

Handbook of Clouds and Big Data

Theory and Practice

Gregor von Laszewski
Geoffrey C. Fox
Judy Qiu

laszewski@gmail.com

Sunday 8th April, 2018 14:05

<http://cyberaide.org/papers/vonLaszewski-bigdata.pdf>

Copyright © 2017, 2018
Gregor von Laszewski
Geoffrey C. Fox
Judy Qiu

laszewski@gmail.com

Originally published at

<https://github.com/cloudmesh/classes>

<http://cyberaide.org/papers/vonLaszewski-bigdata.pdf>

First printing by Gregor von Laszewski, October 2017

Short Table Of Contents

I	Preface	
1	Introduction	59
2	Updates	67
II	Syllabus	
3	Calendar	75
4	E516: Introduction to Cloud Computing	77
5	E616/I524: Advanced Cloud Computing	85
6	E222: Intelligent Systems Engineering II	93
7	Assignments	101

III	Class Policies and Communication	
8	Course Policies	115
9	Piazza	133
IV	Documenting Scientific Research	
10	Documenting Scientific Research	145
11	Introduction to \LaTeX	157
12	Managing Bibliographies	177
13	Editors	195
14	Other Formats	199
V	Development Tools	
15	Linux	209
16	SSH	215
17	Github	225
18	Virtual Box	235
VI	Cloud Resources	
19	FutureSystems	241
20	Chameleon Cloud	247

VII Python

21	Introduction	291
22	Install	293
23	Language	303
24	Data Management	317
25	Libraries	323

VIII Python - Advanced

26	Numpy	339
27	Scipy	343
28	OpenCV	349
29	NIST Pedestrian and Face Detection	359
30	Python Fingerprint Example	371

IX Cloudmesh

31	Cloudmesh Command Shell	387
32	Draft: Enhanced Cloudmesh	391

X Cloud Computing

33	Overview	399
-----------	-----------------------	------------

XI	Cloud Server Technologies	
34	REST	407
35	MQTT	439
36	GraphQL	447
XII	Cloud Container Technologies	
37	Introduction to Containers	451
38	Running Docker Locally	455
39	Docker and Docker Swarm on FutureSystems	459
40	Introduction to Kubernetes	469
41	Apache Hadoop using Docker	477
42	Apache Hadoop Latest (3.0.1) using Docker	485
43	Exercises	495
44	Docker Ecosystem	497
XIII	Cloud Virtual Machine Technologies	
45	Introduction to Virtual Machines	505
XIV	MapReduce	
46	Hadoop	513
47	Spark	519
48	Draft: Harp	537

49	Draft: Twister	539
----	-----------------------------	-----

XV	Draft: Cloud Data Management
-----------	-------------------------------------

50	Data Formats	549
----	---------------------------	-----

51	NoSQL	551
----	--------------------	-----

XVI	Computing with Raspberry Pi
------------	------------------------------------

52	Operating Systems	559
----	--------------------------------	-----

53	PI Software	561
----	--------------------------	-----

54	Computing	565
----	------------------------	-----

XVII	Cloud Computing with Raspberry Pi
-------------	--

55	Pi Cluster Form Factor	571
----	-------------------------------------	-----

56	Cluster Setup	579
----	----------------------------	-----

57	Docker Containers	583
----	--------------------------------	-----

58	Kubernetes	585
----	-------------------------	-----

59	Docker Swarm	587
----	---------------------------	-----

60	Spark	589
----	--------------------	-----

61	Slurm	591
----	--------------------	-----

XVIII	Big Data Applications
--------------	------------------------------

62	Big Data Use Cases Survey	595
----	--	-----

63	Health Informatics	605
----	---------------------------------	-----

64	e-Commerce and LifeStyle	611
65	Physics	617
66	Sensors	625
67	Sports	629
68	Web Search and Text Mining	633

XIX

Cloud and Big Data Technologies

69	Big Data Applications and Software	639
70	Technology Overview	643

XX

Draft: IoT

71	Introduction	763
72	Hardware for IoT Projects	765
73	Projects	769
74	ESP8266	771
75	Raspberry Pi 3	783
76	Dexter	793
77	GrovePi Modules	795
78	VNC	803
79	Turtle Graphics	805
80	Tools	809

XXI	——- Draft Chapters and Parts ——-	
XXII	Draft: Incomming Contributions	
81	Contributed Tutorials	849
XXIII	Draft: Operating System	
XXIV	Draft: PI	
XXV	Draft: Big Data Algorithms	
82	Technology Training - kNN and Clustering	865
83	Technology for Big Data Applications and Analytics	869
84	Technology Training - Plotviz	873
XXVI	Draft: Artificial Intelligence	
XXVI	Outdated: Cloud Computing	
85	OUTDATED: Cloud Computing Fundamentals	883
86	Outdated: IaaS	885
XXVI	Outdated: Data Management	
87	Outdated: NoSQL	891
XXIX	Outdated: SaaS	
88	Outdated: Search Engine	897

XXX **Outdated: MapReduce**

89 **Outdated: MapReduce** 901

90 **Outdated: Iterative Map Reduce** 907

XXXI **Outdated: Internet of Things**

91 **Outdated: IoT** 915

XXXI **Class Projects**

92 **Projects** 919

Bibliography 925

Index 967

Contents

I	Preface	
1	Introduction	59
1.1	Citation	60
1.2	Authors	60
1.3	About	61
1.4	Other Results	61
1.5	Contributing	61
1.5.1	Class Contributions	61
1.5.2	Fixing Typos	62
1.5.3	Community Contributions	62
1.5.4	Contributors	62
1.5.5	Section Contributors	63
1.6	Conventions	63
1.6.1	Videos	63
1.6.2	Slides	63
1.6.3	Images	63
1.6.4	URLs	63
1.6.5	Variables	64
1.6.6	Boxes	64
1.6.7	Copy and Paste	64
1.6.8	Github and Travis and Pull Requests	65
1.6.9	Github Issues	65
1.7	Exercises	65

2	Updates	67
----------	----------------------	-----------

II	Syllabus
-----------	-----------------

3	Calendar	75
----------	-----------------------	-----------

4	E516: Introduction to Cloud Computing	77
----------	--	-----------

4.1	Course Description	77
------------	---------------------------------	-----------

4.2	Course Prerequisites	77
------------	-----------------------------------	-----------

4.3	Registration Information	77
------------	---------------------------------------	-----------

4.4	Teaching and Learning Methods	78
------------	--	-----------

4.5	Covered Topics	78
------------	-----------------------------	-----------

4.6	Student learning outcomes	78
------------	--	-----------

4.7	Grading	79
------------	----------------------	-----------

4.8	Representative bibliography	79
------------	--	-----------

4.9	Lectures and Lecture Material	79
------------	--	-----------

4.9.1	Class Graph E516	79
-------	------------------------	----

4.9.2	Assignments	81
-------	-------------------	----

4.9.3	Communication Track	81
-------	---------------------------	----

4.9.4	Theory Track	81
-------	--------------------	----

4.9.5	Programming Track	82
-------	-------------------------	----

5	E616/I524: Advanced Cloud Computing	85
----------	--	-----------

5.1	Course Description	85
------------	---------------------------------	-----------

5.2	Course pre-requisites	85
------------	------------------------------------	-----------

5.3	Course Registration	85
------------	----------------------------------	-----------

5.4	Teaching and learning methods	86
------------	--	-----------

5.5	Covered Topics	86
------------	-----------------------------	-----------

5.6	Student learning outcomes	87
------------	--	-----------

5.7	Grading	87
------------	----------------------	-----------

5.8	Representative bibliography	87
------------	--	-----------

5.9	Lectures and Lecture Material	88
------------	--	-----------

5.9.1	Class Graph E616 and I524	88
-------	---------------------------------	----

5.9.2	Assignments	88
-------	-------------------	----

5.9.3	Communication Track	88
-------	---------------------------	----

5.9.4	Theory Track	90
-------	--------------------	----

5.9.5	Programming Track	91
-------	-------------------------	----

5.9.6	DevOps	91
-------	--------------	----

5.9.7	Cloud	91
-------	-------------	----

5.10	Containers	92
-------------	-------------------------	-----------

6	E222: Intelligent Systems Engineering II	93
----------	---	-----------

6.1	Course Description	93
------------	---------------------------------	-----------

6.2	Course pre-requisites	93
------------	------------------------------------	-----------

6.3	Course Registration	93
6.4	Teaching and learning methods	93
6.5	Covered Topics	93
6.6	Student learning outcomes	94
6.7	Grading	94
6.8	Representative bibliography	94
6.9	Lectures and Lecture Material	95
6.9.1	Class Graph E516	95
6.9.2	Proposed Weekly Agenda	95
6.9.3	Week 1. Administration	95
6.9.4	Weeks 1, 2 and 4. Introduction to Cloud Computing	95
6.9.5	Week 3. REST for Cloud computing	97
6.9.6	Weeks 2-3. Setting up your development environment	98
6.9.7	Week 5. Introduction to simple Containers	98
6.9.8	Week 6. Introduction to Container Clusters	98
6.9.9	Week 7. Map Reduce	98
6.9.10	Week 8. Overview of Cloud Services	99
6.9.11	Week 9. Multi cloud environments	99
6.9.12	Week 10. Cloud Data and Applications	99
6.9.13	Other weeks	99
6.9.14	Assignments	99
6.9.15	Communication Track	99
6.9.16	Theory Track	100
6.9.17	Programming Track	100
7	Assignments	101
7.1	Assignments E222	102
7.1.1	Bio Post	102
7.1.2	Cloud Accounts	102
7.1.3	Account Creation	102
7.1.4	Entry Survey	103
7.1.5	Account Setup	103
7.1.6	Eve REST Service	103
7.2	Assignments E516, I524, E616	104
7.2.1	Bio Post	104
7.2.2	IU Google Services	104
7.2.3	Big Data Collaboration	104
7.2.4	New Technology List	105
7.2.5	New Technology Abstract	105
7.2.6	Cloud Accounts	106
7.2.7	Entry Survey	106
7.2.8	Account Setup	106
7.2.9	REST	107
7.2.10	Swagger REST Services	107
7.2.11	Technology Paper	110
7.2.12	Tutorial	110
7.2.13	Project	111

8	Course Policies	115
8.1	Communication and Use of CANVAS	115
8.2	Managing Your Own Calendar	115
8.3	Online and Office Hours	116
8.3.1	Office Hour Calendar	116
8.4	Class Material	117
8.5	HID	117
8.6	Notebook	117
8.7	Blog	118
8.8	Calendar I524, E616, E516	118
8.9	Incomplete	118
8.10	Waitlist	119
8.11	Registration	120
8.12	Auditing the class	120
8.13	Resource restrictions	121
8.14	Plagiarism	121
8.15	Tutorials, Topic Paper, Term Paper, Project Report	122
8.15.1	Team	123
8.15.2	Common Deliverables	123
8.15.3	Project Paper	124
8.15.4	Term Paper	125
8.16	Project Ideas	126
8.16.1	Project Data Restrictions	126
8.16.2	Example Projects	126
8.16.3	Register your project idea	126
8.16.4	Meeting with Gregor	126
8.16.5	Project type A: NIST Rest services project	126
8.16.6	Project type B: Raspberry PI projects	127
8.16.7	Project type C: Data related project for Spark or Hadoop	128
8.16.8	Project type D: Data related project for a kubernetes or swarm cluster	128
8.16.9	Project type E: Define your own	128
8.16.10	Project Idea Piazza Notes	128
8.16.11	Docker Cluster on PI Video	129
8.16.12	Hadoop 3.0	129
8.17	Grading	131
8.17.1	Grades on Canvas	131
8.17.2	Discussion about Grades	131
9	Piazza	133
9.1	Access to Piazza from Canvas	133
9.2	Verify you are on Piazza via a post	136
9.3	Making Piazza Work	136

9.4	Towards good questions	136
9.5	Guide on how to ask good questions	137
9.6	Piazza class Links	138
9.6.1	Current Classes	138
9.6.2	Previous Classes	138
9.7	Piazza Curation	139
9.8	Read the Originals, not just the e-mail	140
9.9	Exercises	140
9.10	Fall 2018	141
9.10.1	E534	141
9.10.2	I523	141
9.10.3	I423	141

IV

Documenting Scientific Research

10	Documenting Scientific Research	145
10.1	Overview	145
10.2	Plagiarism	146
10.2.1	Plagiarism Definition	146
10.2.2	Plagiarism Policies	146
10.2.3	Plagiarism Resources	147
10.2.4	Tutorials	147
10.2.5	How to Recognize Plagiarism	147
10.3	Acknowledgements	148
10.4	Writing a Scientific Article or Conference Paper	149
10.4.1	Professional Paper Format	149
10.4.2	Submission Requirements	150
10.4.3	Microsoft Word vs. \LaTeX	150
10.4.4	Working in a Team	151
10.4.5	Time Management	151
10.4.6	Paper and Report Checklist	151
10.4.7	Content	152
10.4.8	Submission	152
10.4.9	Bibliography	152
10.4.10	Writing	153
10.4.11	Citation Issues and Plagiarism	153
10.4.12	Character Errors	153
10.4.13	Structural Issues	154
10.4.14	Figures and Tables	154
10.4.15	Example Paper	155
10.4.16	Creating the PDF from \LaTeX on your Computer	155
10.4.17	Draft: Class Specific README.md	155
10.4.18	Exercises	156

11	Introduction to \LaTeX	157
11.1	Installation	157
11.1.1	Local Install	157
11.1.2	Online Services	158
11.2	Basic \LaTeX Elements	160
11.2.1	Characters	160
11.2.2	Highlighting Text	160
11.2.3	Sections	160
11.2.4	Empty Lines	161
11.2.5	Itemize	161
11.2.6	Enumerate	161
11.2.7	Descriptions	161
11.2.8	Images	162
11.2.9	Tables	162
11.2.10	Labels	164
11.2.11	Mathematics	164
11.3	Advanced topics	165
11.3.1	ACM and IEEE Proceedings Format	165
11.3.2	Generating and Managing Images	165
11.3.3	Colored Boxes	166
11.3.4	Slides	167
11.3.5	\LaTeX vs. X	167
11.4	Editing	168
11.4.1	Emacs	168
11.4.2	Vi/Vim	169
11.4.3	TeXshop	169
11.4.4	LyX	169
11.4.5	WYSIWYG locally	170
11.4.6	Markdown and \LaTeX	170
11.4.7	Including RST into \LaTeX	171
11.4.8	pyCharm	171
11.4.9	MSWord	171
11.5	The \LaTeX Cycle	171
11.6	Tips	172
11.6.1	Colorful Output	172
11.6.2	latex2html	173
11.6.3	lachek	173
11.6.4	chktex	174
11.6.5	Chacking For Non-ASCII Charaters	175
11.6.6	References	175
12	Managing Bibliographies	177
12.1	Integrating Bibliographies	177
12.1.1	biber	177
12.1.2	bibtex	177
12.1.3	jabref	178
12.2	Entry types	178
12.2.1	Source Code References	179

12.2.2	Pedigree	181
12.2.3	Article in a Journal	181
12.2.4	Article in a Conference Proceedings	183
12.2.5	InProceedings	186
12.2.6	TechReport	186
12.2.7	Article	187
12.2.8	Proceedings	187
12.2.9	Wikipedia Entry	188
12.2.10	Blogs	188
12.2.11	Web Page	189
12.2.12	Book	189
12.3	Integrating Bibtex entries into Other Systems	191
12.3.1	jabref and MSWord	192
12.3.2	Bibtex import to MSWord	192
12.4	Other Reference Managers	192
12.4.1	Endnote	192
12.4.2	Mendeley	192
12.4.3	Zotero	193
12.4.4	Paperpile	193
13	Editors	195
13.1	Basic Emacs	195
13.1.1	Org Mode	197
13.1.2	Programming Python with Emacs	198
13.1.3	Emacs Keys in a Terminal	198
13.1.4	LaTeX and Emacs	198
14	Other Formats	199
14.1	reStructuredText	199
14.1.1	Links	199
14.1.2	Source	199
14.1.3	Sections	200
14.1.4	Listtable	200
14.1.5	Excelltable	200
14.1.6	Boxes	200
14.1.7	Sidebar directive	201
14.1.8	Sphinx Prompt	201
14.1.9	Programm examples	201
14.1.10	Hyperlinks	201
14.1.11	Todo	202
14.2	Markdown	202
14.2.1	Tools	203
14.2.2	Presentations in Markdown	203
14.3	Communicating Research in Other Ways	203
14.3.1	Blogs	203
14.3.2	Sphinx	204
14.3.3	Notebooks	204

15	Linux	209
15.1	History	209
15.2	Shell	209
15.3	Multi-command execution	212
15.4	Keyboard Shortcuts	212
15.5	bashrc and bash_profile	213
15.6	Makefile	213
15.7	Exercises	214
16	SSH	215
16.1	Generate a SSH key	215
16.2	Add or Replace Passphrase	216
16.3	SSH Add and Agent	217
16.4	SSH Config	217
16.5	SSH Port Forwarding	217
16.6	Upload the key to gitlab	217
16.7	Using SSH on Mac OS X	218
16.8	Using SSH on Linux	218
16.9	Using SSH on Raspberry Pi 3	218
16.10	SSH on Windows	218
16.10.1	OpenSSH Client (Beta)	218
16.10.2	Using SSH from Cygwin	219
16.10.3	SSH from putty	219
16.10.4	Chocolatey	220
16.11	Tips	221
16.12	SSH to FutureSystems Resources	222
16.13	Testing your ssh key	223
16.14	Refernces	223
16.15	Exercises	223
17	Github	225
17.1	Obverview	225
17.2	Upload Key	226
17.3	Fork	226
17.4	Rebase	226
17.5	Remote	226
17.6	Pull Request	227
17.7	Branch	227
17.8	Checkout	227
17.9	Merge	227

17.10	GUI	227
17.11	Windows	228
17.12	Git from the Commandline	228
17.13	Configuration	228
17.14	Upload your public key	229
17.15	Working with a directory that was provided for you	229
17.16	README.yml and notebook.md	230
17.17	Contributing to the Document	231
17.17.1	Clone	231
17.17.2	Merge	231
17.17.3	Resources	232
17.18	Exercises	232
17.19	Github Issues	232
17.19.1	Git Issue Features	233
17.19.2	Github Markdown	233
17.19.3	Notifications	234
17.19.4	cc	234
17.19.5	Interacting with issues	234
18	Virtual Box	235
18.1	Instalation	235
18.2	Guest additions	236
18.3	Exercises	237

VI

Cloud Resources

19	FutureSystems	241
19.1	FutureSystems evolved from FutureGrid	241
19.2	Creating Portal Account	241
19.3	SSH Key Generation using ssh-keygen command	241
19.4	Shell Access via SSH	242
19.5	Advanced SSH	242
19.6	SSH Key Generation via putty	242
19.7	FutureSystems Facilities	242
19.7.1	Bravo	242
19.7.2	Delta	243
19.7.3	Echo	243
19.7.4	Juliet	243
19.7.5	Romeo	243
19.7.6	Tango	243
19.7.7	Tempest	244
19.7.8	Victor	244
19.7.9	PI Cluster	244

20	Chameleon Cloud	247
20.1	Overview	247
20.2	Resources	248
20.3	Hardware	249
20.3.1	Standard Cloud Units	249
20.3.2	Network	249
20.3.3	Shared Storage	250
20.3.4	Heterogeneous Compute Hardware	250
20.3.5	Live updates	251
20.4	Getting Started	251
20.4.1	Step 1: Create a Chameleon account	251
20.4.2	Step 2: Create or join a project	252
20.4.3	Step 3: Start using Chameleon!	252
20.5	Charge Rates	253
20.5.1	Service Units	253
20.5.2	Project Allocation Size	254
20.6	OpenStack Virtual Machines	254
20.6.1	Web Interface	255
20.6.2	OpenStack REST Interfaces	260
20.6.3	Downloading and uploading data	261
20.7	Horizon Graphical User Interface	261
20.7.1	Configure resources	261
20.7.2	Interact with resources	266
20.7.3	Use FPGAs	268
20.7.4	Next Step	268
20.8	HEAT	268
20.8.1	Supporting Complex Appliances	268
20.8.2	Chameleon Appliance Catalog	269
20.8.3	Deployment	269
20.8.4	Heat Template	272
20.8.5	Customizing an existing template	275
20.8.6	Writing a new template	279
20.8.7	Sharing new complex appliances	280
20.8.8	Advanced topics	281
20.9	Bare Metal	284
20.10	Frequently Asked Questions	284
20.10.1	Appliances	285
20.10.2	Bare Metal Troubleshooting	286
20.10.3	OpenStack KVM Troubleshooting	287

VII

Python

21	Introduction	291
21.1	Introduction to Python	291
21.2	References	292

22	Install	293
22.1	Python Installation	293
22.1.1	Managing custom Python installs	293
22.1.2	Updating Python Version List	297
22.1.3	Updating to a new version of Python with pyenv	297
22.1.4	Installation without pyenv	297
22.1.5	Anaconda and Miniconda	298
22.2	Interactive Python	300
22.3	REPL (Read Eval Print Loop)	300
22.4	Python 3 Features in Python 2	301
23	Language	303
23.1	Statements and Strings	303
23.2	Variables	303
23.3	Data Types	304
23.3.1	Booleans	304
23.3.2	Numbers	304
23.4	Module Management	305
23.4.1	Import Statement	305
23.4.2	The from ... import Statement	305
23.5	Date Time in Python	306
23.6	Control Statements	307
23.6.1	Comparison	307
23.6.2	Iteration	308
23.7	Datatypes	308
23.7.1	Lists	308
23.7.2	Sets	310
23.7.3	Removal and Testing for Membership in Sets	310
23.7.4	Dictionaries	311
23.7.5	Dictionary Keys and Values	312
23.7.6	Counting with Dictionaries	312
23.8	Functions	313
23.9	Classes	313
23.10	Modules	314
23.11	Lambda Expressions	315
23.12	Generators	316
23.13	Non Blocking Threads	316
24	Data Management	317
24.1	Formats	317
24.1.1	Pickle	317
24.1.2	Text Files	317
24.1.3	CSV Files	318
24.1.4	Excel spread sheets	318
24.1.5	YAML	318

24.1.6	JSON	319
24.1.7	XML	319
24.1.8	RDF	319
24.1.9	PDF	319
24.1.10	HTML	320
24.1.11	ConfigParser	320
24.1.12	ConfigDict	320
24.2	Encryption	320
24.3	Database Access	321
24.4	SQLite	321
24.4.1	Exercises	321
25	Libraries	323
25.1	Installing Libraries	323
25.2	Using pip to Install Packages	323
25.3	GUI	324
25.3.1	GUIZero	324
25.3.2	Kivy	324
25.4	Formatting and Checking Python Code	324
25.5	Using autopep8	324
25.6	Writing Python 3 Compatible Code	325
25.7	Using Python on FutureSystems	325
25.8	Ecosystem	325
25.8.1	pypi	325
25.8.2	Alternative Installations	325
25.9	Resources	327
25.9.1	Jupyter Notebook Tutorials	327
25.10	Exercises	327
25.11	Python for Big Data	328
25.11.1	An Example with Pandas, NumPy and Matplotlib	328
25.11.2	Summary of Useful Libraries	333
25.11.3	Big Data Libraries	333
25.12	Parsing Data	335
25.12.1	notebook.md Parser	335
25.12.2	Video Length	336
25.12.3	Dask	336

VIII

Python - Advanced

26	Numpy	339
26.1	Float Range	339
26.2	Arrays	340
26.3	Array Operations	340
26.4	Linear Algebra	341

26.5	Resources	341
27	Scipy	343
27.1	Introduction	343
27.2	References	347
28	OpenCV	349
28.1	Overview	349
28.2	Installation	349
28.3	A Simple Example	350
28.3.1	Loading an image	350
28.3.2	Displaying the image	350
28.3.3	Scaling and Rotation	350
28.3.4	Gray-scaling	351
28.3.5	Image Thresholding	351
28.3.6	Edge Detection	351
28.4	Additional Features	352
28.5	Secchi Disk	353
28.5.1	Overview	353
28.6	Setup for OSX	353
28.6.1	Step 1: Record the video	353
28.6.2	Step 2: Analyse the images from the Video	353
28.7	Black and white	358
29	NIST Pedestrian and Face Detection	359
29.0.1	Introduction	360
29.0.2	Deployment by Ansible	361
29.0.3	Cloudmesh for Provisioning	362
29.0.4	Roles Explained for Installation	362
29.0.5	Instructions for Deployment	363
29.0.6	OpenCV in Python	364
29.0.7	Human and Face Detection in OpenCV	366
29.0.8	Pedestrian Detection using HOG Descriptor	368
29.0.9	Processing by Apache Spark	369
29.0.10	Results for 100+ images by Spark Cluster	370
30	Python Fingerprint Example	371
30.1	Overview	371
30.2	Objectives	372
30.3	Prerequisites	372
30.4	Implementation	372
30.5	Utility functions	373
30.6	Dataset	374
30.7	Data Model	375
30.7.1	Utilities	375
30.7.2	Checksum	375

30.7.3	Path	376
30.7.4	Image	376
30.7.5	Mindtct	376
30.7.6	Bozorth3	377
30.7.7	Running Bozorth3	378
30.8	Plotting	379
30.9	Putting it all Together	379

IX

Cloudfmesh

31	Cloudfmesh Command Shell	387
31.1	CMD5	387
31.1.1	Resources	387
31.1.2	Creating a Python Development Environment	387
31.1.3	Installation from source	387
31.1.4	Execution	388
31.1.5	Create your own Extension	388
31.1.6	Exercises	389
32	Draft: Enhanced Cloudfmesh	391
32.1	Configuration	392
32.2	Storage	393
32.2.1	sqlite3	393
32.2.2	Context	394

X

Cloud Computing

33	Overview	399
33.1	Introduction to Cloud Computing	400
33.1.1	Introduction - Part A	400
33.1.2	Introduction - Part B - Defining Clouds I	400
33.1.3	Introduction - Part C - Defining Clouds II	401
33.1.4	Introduction - Part D - Defining Clouds III	401
33.1.5	Introduction - Part E - Virtualization	401
33.1.6	Introduction - Part F - Technology Hypecycle I	401
33.1.7	Introduction - Part G - Technology Hypecycle II	401
33.1.8	Introduction - Part H - IaaS I	402
33.1.9	Introduction - Part I - IaaS II	402
33.1.10	Introduction - Part J - Cloud Software	402
33.1.11	Introduction - Part K - Applications I	402
33.1.12	Introduction - Part L - Applications II	403
33.1.13	Introduction - Part M - Applications III	403
33.1.14	Introduction - Part N - Parallelism	403
33.1.15	Introduction - Part O - Storage	403
33.1.16	Introduction - Part P - HPC in the Cloud	403
33.1.17	Introduction - Part Q - Analytics and Simulation	403
33.1.18	Introduction - Part R - Jobs	404

33.1.19	Introduction - Part S - The Future	404
33.1.20	Introduction - Part T - Security	404
33.1.21	Introduction - Part U - Fault Tolerance	404

XI

Cloud Server Technologies

34	REST	407
34.1	Overview of REST	407
34.1.1	Collection of Resources	408
34.1.2	Single Resource	408
34.1.3	REST Tool Classification	408
34.2	Flask RESTful Services	408
34.3	Rest Services with Eve	410
34.3.1	Ubuntu install of MongoDB	411
34.3.2	OSX install of MongoDB	411
34.3.3	Windows 10 Installation of MongoDB	411
34.3.4	Database Location	411
34.3.5	Verification	411
34.3.6	Building a simple REST Service	412
34.3.7	Interacting with the REST service	413
34.3.8	Creating REST API Endpoints	416
34.3.9	REST API Output Formats and Request Processing	417
34.3.10	WRONG: Consuming REST API Using a Client Application	418
34.3.11	HATEOAS	419
34.3.12	Pretty Printing	420
34.3.13	XML	420
34.3.14	Extensions to Eve	420
34.4	Object Management with Eve and Evegenie	421
34.4.1	Installation	421
34.4.2	Starting the service	422
34.4.3	Creating your own objects	422
34.5	Towards cmd5 extensions to manage eve and mongo	423
34.6	Responses	423
34.7	Django REST Framework	423
34.8	Rest Services with Swagger	424
34.9	Swagger Tools	424
34.10	Swagger Community Tools	424
34.10.1	Swagger Toolbox	424
34.11	REST Service Generation with Swagger	426
34.11.1	Step 1: Define Your REST Service	426
34.11.2	Step 2: Server Side Stub Code Generation and Implementation	427
34.11.3	Step 3: Install and Run the REST Service:	429
34.11.4	Step 4: Generate Client Side Code and Verify	430
34.11.5	Towards a Distributed Client Server	431
34.11.6	Exercises	431

34.12	Swagger Specification	433
34.12.1	The Virtual Cluster example API Definition	434
34.12.2	Refernces	437
35	MQTT	439
35.1	Introduction	439
35.2	Publish Subscribe Model	440
35.2.1	Topics	440
35.2.2	Callbacks	440
35.2.3	Quality of Service	441
35.3	Secure MQTT Services	441
35.3.1	Using TLS/SSL	441
35.3.2	Using OAuth	442
35.4	Integration with Other Services	442
35.5	MQTT in Production	442
35.6	Simple Usecase	442
35.7	IoT Use Case	443
35.7.1	Requirements and Setup	443
35.7.2	Results	444
35.8	Conclusion	444
35.9	Exercises	444
36	GraphQL	447

XII

Cloud Container Technologies

37	Introduction to Containers	451
37.1	Motivation - Microservices	451
37.2	Motivation - Serverless Computing	451
37.3	Docker	451
37.4	Docker and Kubernetes	452
37.5	QEMU and KVM	452
37.6	Resources	452
37.6.1	Container Orchestration Tools: Compare Kubernetes vs Docker Swarm	452
37.6.2	Gentle introduction to Containers	452
37.6.3	Tutorialspoint	452
37.7	Intorduction to Docker	452
37.8	Docker Survey	452
38	Running Docker Locally	455
38.1	Instillation for OSX	455
38.2	Installation for Ubuntu	456
38.3	Draff: Installation for Windows 10	456

38.4	Testing the Install	457
39	Docker and Docker Swarm on FutureSystems	459
39.1	Getting Access	459
39.2	Creating a service and deploy to the swarm cluster	460
39.2.1	Create your own service	461
39.2.2	Exercises	461
39.3	Dockerized REST Service	461
39.3.1	Prerequisites	462
39.3.2	Activate Virtual Environment	462
39.3.3	File Structure	463
39.3.4	Build Docker Image	465
39.3.5	Run Docker Image	465
40	Introduction to Kubernetes	469
40.1	Topics Covered and Learning Outcome	469
40.2	What is Kubernetes?	469
40.3	What are containers?	470
40.4	Terminology	470
40.5	Kubernetes Architecture	471
40.6	Minikube	471
40.6.1	Install minikube	471
40.6.2	Start a cluster using Minikube	472
40.6.3	Create a deployment	472
40.6.4	Expose the service	472
40.6.5	Check running status	472
40.6.6	Call service api	473
40.6.7	Take a look from Dashboard	473
40.6.8	Delete the service and deployment	473
40.6.9	Stop the cluster	473
40.7	Interactive Tutorial Online	473
40.8	Using Kubernetes on FutureSystems	473
40.8.1	Getting Access	473
40.8.2	Example Use	474
40.8.3	Exercises	475
41	Apache Hadoop using Docker	477
41.1	Draft: Creating the Hadoop Container	477
41.2	Hadoop from Docker	477
41.3	Start a Hadoop container	477
41.4	Statistical Example with Hadoop	477
41.4.1	Description	478
41.4.2	Base Location	478
41.4.3	Input Files	478
41.4.4	Compilation	478
41.4.5	Archiving Class Files	479

41.4.6	HDFS for Input/Output	479
41.4.7	Run Program with a Single Input File	479
41.4.8	Result for Single Input File	481
41.4.9	Run Program with Multiple Input Files	481
41.4.10	Result for Multiple Files	483
41.5	Conclusion	483
42	Apache Hadoop Latest (3.0.1) using Docker	485
42.1	Draft: Building Hadoop 3.0.1 using Docker	485
42.2	Start a Hadoop Container with Interactive Shell	485
42.3	Examples	486
42.4	Draft: Apache Spark with Docker	486
42.4.1	Pull Image from Docker Repository	486
42.4.2	Running the Image	486
42.4.3	Run Spark	486
42.4.4	Observe Task Execution from Running Logs of SparkPi	486
42.4.5	Write a Word-Count Application with Spark RDD	487
42.4.6	Docker Spark Examples	487
42.4.7	Interactive Examples	488
42.5	Draft: Multinode Hadoop Cluster Deployment with Docker Swarm	493
42.5.1	Docker Hadoop 3.0.1	493
43	Exercises	495
43.1	Docker Swarm Tutorial	495
43.2	Docker Swarm on Google Compute Engine	495
43.3	Single Node Hadoop	495
43.4	Single Node Hadoop	496
43.5	Spark Cluster	496
44	Docker Ecosystem	497
44.1	Docker Hub	497
44.1.1	Create Docker ID and Log In	497
44.1.2	Searching for Docker Images	498
44.1.3	Pulling Images	498
44.1.4	Create Repositories	498
44.1.5	Pushing Images	499
44.1.6	Resources	500

XIII

Cloud Virtual Machine Technologies

45	Introduction to Virtual Machines	505
45.1	QEMU and KVM	505
45.2	Chameleon OpenStack	506
45.2.1	Outages	506
45.2.2	Account Creation	506

45.2.3	Join a Project	506
45.2.4	Usage Restriction	507
45.2.5	OpenStack RC File	507
45.2.6	CLI to Manage Virtual Machines	508
45.2.7	Creating SSH keys	509
45.2.8	KeyPair Registration	509
45.2.9	Start a new VM instance	509
45.2.10	Floating IP Address	509
45.2.11	Termination of VM Instance	510

XIV

MapReduce

46	Hadoop	513
46.1	Hadoop Motivation	513
46.2	Hadoop and MapReduce	513
46.3	Hadoop EcoSystem	513
46.4	Hadoop Components	514
46.5	Hadoop and the Yarn Resource Manager	514
46.6	PageRank	514
46.7	Installation of Hadoop	515
46.7.1	Prerequisites	515
46.7.2	User and User Group Creation	515
46.7.3	Configuring SSH	515
46.7.4	Installation of Java	516
46.7.5	Installation of Hadoop	516
46.7.6	Hadoop Environment Variables	516
47	Spark	519
47.1	Motivation for Spark	519
47.2	Spark RDD Operations	519
47.3	Spark DAG	519
47.4	Spark vs. other Frameworks	520
47.5	Installation of Spark	521
47.5.1	Prerequisites	521
47.5.2	Installation of Java	521
47.5.3	Install Spark with Hadoop	521
47.5.4	Spark Environment Variables	522
47.5.5	Test Spark Installation	522
47.5.6	Install Spark With Custom Hadoop	522
47.5.7	Configuring Hadoop	523
47.5.8	Test Spark Installation	523
47.6	Spark Streaming	525
47.6.1	Streaming Concepts	525
47.6.2	Simple Streaming Example	525
47.6.3	Spark Streaming For Twitter Data	526

48	Draft: Harp	537
48.1	Harp	538
49	Draft: Twister	539
49.1	Twister2 Examples	540
49.1.1	Introduction to Twister2	540
49.1.2	Twister2 Examples	540
49.2	Twister2 Installation	541
49.2.1	Prerequisites	541
49.2.2	Compiling Twister2	542

XV

Draft: Cloud Data Management

50	Data Formats	549
50.1	YAML	549
50.2	JSON	550
50.3	XML	550
51	NoSQL	551
51.1	RDBMS vs. NoSQL	551
51.2	NoSQL Characteristics	551
51.2.1	Document Model	552
51.2.2	Graph Model	552
51.2.3	Key-Value and Wide Column Models	552
51.3	BigTable	552
51.4	HBase	553
51.5	HBase Coding	553
51.6	Draft: MongoDB	553
51.7	Indexing Applications	553
51.8	Related Work	554
51.9	Indexexamples	554
51.10	Indexing 101	554
51.11	Social Media Searches	555
51.12	Analysis Algorithms	555

XVI

Computing with Raspberry Pi

52	Operating Systems	559
52.0.1	Operating Systems	559

53	PI Software	561
53.1	Editors	561
53.1.1	emacs	561
53.1.2	vim	561
53.1.3	gedit	561
53.1.4	ssh	561
53.2	Python 3	561
53.3	Python IDLE	562
53.4	Docker	562
53.5	Go	562
53.5.1	Eclipse	562
53.5.2	Adafruit Web IDE	562
53.5.3	Coder	562
54	Computing	565
54.1	Numpy	565
54.2	Scipy	565
54.3	Image Processing	565
54.4	DHCP Server	565
54.5	Gregor's Notes	565
54.5.1	editor	566
54.5.2	hostname	566
54.5.3	Gather the mac addresses	566
54.5.4	Enable sshd	566
54.5.5	Wireless	567
54.5.6	Update when on network	567
54.5.7	USB stick	567
54.5.8	Locale	567
54.5.9	DHCP server on 00	567
54.5.10	Temperature	568
54.5.11	graphana	568

XVII Cloud Computing with Raspberry Pi

55	Pi Cluster Form Factor	571
55.1	NAS (1 Pi)	571
55.2	ClusterHat (4 Zero + 1 Pi)	572
55.3	Cluster Case With Cooling (5 Pi)	572
55.4	Bitscope Case (40 Pi)	573
55.5	Bitscope Cluster (144 Pi)	573
55.5.1	Links	574
55.6	Build Your Own 5 Node Pi Cluster	574
55.7	Single Pi	577

55.8	Small Pi Cluster	577
55.8.1	Virtual Raspberry Cluster	578
56	Cluster Setup	579
56.1	Links	579
56.2	SDCards	580
56.2.1	Linux	580
56.2.2	OSX	580
56.2.3	Windows 10	580
56.2.4	Creating Backup	580
56.2.5	Duplication	580
56.2.6	Mount the SD Card on the Host System	580
56.2.7	Linux	581
56.2.8	OSX	581
56.2.9	Windows 10	581
56.3	System Preparation without Monitor	581
56.3.1	hostname	581
56.3.2	SSH	581
56.3.3	key	581
56.3.4	password	581
56.4	Post confoguration	581
56.4.1	Network Addresses	581
56.4.2	key	582
56.4.3	VNC	582
56.5	Addon Hardware	582
57	Docker Containers	583
57.1	Instalation	583
57.2	Configuration	583
57.3	Example Container Creation with Dockerfile	583
57.4	Using the Container	583
57.5	REST Services	583
58	Kubernetes	585
58.1	Instalation	585
58.2	Configuration	585
58.3	Example Container Creation with Dockerfile	585
58.4	Using the Container in Kubernetes	585
58.5	REST Services	585
59	Docker Swarm	587
59.1	Instalation	587
59.2	Configuration	587
59.3	Example Swarm program	587
59.4	Using the Swarm	587

59.5	REST Services	587
60	Spark	589
60.1	Instalation	589
60.2	Configuration	589
60.3	Example Spark program	589
60.4	Using Spark	589
60.5	REST Services	589
61	Slurm	591
61.1	Instalation	591
61.2	Configuration	591
61.3	Example MPI program	591
61.4	Using the Batch Queue	591
61.5	REST Services	591

XVIII **Big Data Applications**

62	Big Data Use Cases Survey	595
62.1	NIST Big Data Public Working Group	595
62.1.1	Introduction to NIST Big Data Public Working	595
62.1.2	Definitions and Taxonomies Subgroup	596
62.1.3	Reference Architecture Subgroup	596
62.1.4	Security and Privacy Subgroup	596
62.1.5	Technology Roadmap Subgroup	596
62.1.6	Interfaces Subgroup	597
62.1.7	Requirements and Use Case Subgroup	597
62.2	51 Big Data Use Cases	598
62.2.1	Government Use Cases	598
62.2.2	Commercial Use Cases	598
62.2.3	Defense Use Cases	598
62.2.4	Healthcare and Life Science Use Cases	598
62.2.5	Deep Learning and Social Networks Use Cases	599
62.2.6	Research Ecosystem Use Cases	599
62.2.7	Astronomy and Physics Use Cases	599
62.2.8	Environment, Earth and Polar Science Use Cases	599
62.2.9	Energy Use Case	599
62.3	Features of 51 Big Data Use Cases	600
62.3.1	Summary of Use Case Classification	600
62.3.2	Database(SQL) Use Case Classification	600
62.3.3	NoSQL Use Case Classification	600
62.3.4	Other Use Case Classifications	600
62.3.5	Resources	601

63	Health Informatics	605
63.1	Big Data and Health	605
63.2	Status of Healthcare Today	606
63.3	Telemedicine (Virtual Health)	606
63.4	Medical Big Data in the Clouds	606
63.4.1	Medical image Big Data	606
63.4.2	Clouds and Health	606
63.4.3	McKinsey Report on the big-data revolution in US health care	606
63.4.4	Microsoft Report on Big Data in Health	606
63.4.5	EU Report on Redesigning health in Europe for 2020	607
63.4.6	Medicine and the Internet of Things	607
63.4.7	Extrapolating to 2032	607
63.4.8	Genomics, Proteomics and Information Visualization	607
63.4.9	Resources	608
64	e-Commerce and LifeStyle	611
64.1	Recommender Systems	612
64.1.1	Recommender Systems as an Optimization Problem	612
64.1.2	Recommender Systems Introduction	612
64.1.3	Kaggle Competitions	612
64.1.4	Examples of Recommender Systems	612
64.1.5	Netflix on Recommender Systems	613
64.1.6	Other Examples of Recommender Systems	613
64.1.7	Resources	614
64.2	Item-based Collaborative Filtering and its Technologies	614
64.2.1	Item-based Collaborative Filtering	614
64.2.2	k-Nearest Neighbors and High Dimensional Spaces	615
64.2.3	Resources	615
65	Physics	617
65.1	Looking for Higgs Particles	617
65.1.1	Bumps in Histograms, Experiments and Accelerators	617
65.1.2	Particle Counting	618
65.1.3	Experimental Facilities	618
65.1.4	Accelerator Picture Gallery of Big Science	618
65.1.5	Resources	618
65.1.6	Event Counting	619
65.1.7	Monte Carlo	619
65.1.8	Resources	619
65.1.9	Random Variables, Physics and Normal Distributions	619
65.1.10	Statistics Overview and Fundamental Idea: Random Variables	619
65.1.11	Physics and Random Variables	619
65.1.12	Statistics of Events with Normal Distributions	620
65.1.13	Gaussian Distributions	620
65.1.14	Using Statistics	620
65.1.15	Resources	620
65.1.16	Random Numbers, Distributions and Central Limit Theorem	620
65.2	SKA – Square Kilometer Array	621

65.3	Radar	622
65.3.1	Introduction	622
65.3.2	Remote Sensing	622
65.3.3	Ice Sheet Science	622
65.3.4	Global Climate Change	623
65.3.5	Radio Overview	623
65.3.6	Radio Informatics	623
66	Sensors	625
66.1	Internet of Things	625
66.2	Robotics and IoT	625
66.3	Industrial Internet of Things	626
66.4	Sensor Clouds	626
66.5	Earth/Environment/Polar Science data gathered by Sensors	626
66.6	Ubiquitous/Smart Cities	626
66.7	U-Korea (U=Ubiquitous)	626
66.8	Smart Grid	626
66.9	Resources	627
67	Sports	629
67.1	Basic Sabermetrics	629
67.1.1	Introduction and Sabermetrics (Baseball Informatics) Lesson	629
67.1.2	Basic Sabermetrics	630
67.1.3	Wins Above Replacement	630
67.2	Advanced Sabermetrics	630
67.2.1	Pitching Clustering	630
67.2.2	Pitcher Quality	630
67.3	PITCHf/X	630
67.3.1	Other Video Data Gathering in Baseball	630
67.4	Other Sports	630
67.4.1	Wearables	631
67.4.2	Soccer and the Olympics	631
67.4.3	Spatial Visualization in NFL and NBA	631
67.4.4	Tennis and Horse Racing	631
67.4.5	Resources	631
68	Web Search and Text Mining	633
68.1	Web Search and Text Mining	633
68.1.1	The Problem	634
68.1.2	Information Retrieval	634
68.1.3	History	634
68.1.4	Key Fundamental Principles	634
68.1.5	Information Retrieval (Web Search) Components	634
68.2	Search Engines	635
68.2.1	Boolean and Vector Space Models	635
68.2.2	Web crawling and Document Preparation	635

68.2.3	Indices	635
68.2.4	TF-IDF and Probabilistic Models	635
68.3	Topics in Web Search and Text Mining	635
68.3.1	Data Analytics for Web Search	635
68.3.2	Link Structure Analysis including PageRank	636
68.3.3	Web Advertising and Search	636
68.3.4	Clustering and Topic Models	636
68.3.5	Resources	636

XIX

Cloud and Big Data Technologies

69	Big Data Applications and Software	639
69.1	Overview	639
69.2	Introduction	639
69.3	Application Structure	640
69.4	Application Aspects	640
69.5	Applications	640
69.6	Other	640
70	Technology Overview	643
70.1	Workflow-Orchestration	643
70.1.1	ODE	643
70.1.2	ActiveBPEL	643
70.1.3	Airavata	644
70.1.4	Pegasus	644
70.1.5	Kepler	644
70.1.6	Swift	644
70.1.7	Taverna	645
70.1.8	Triana	645
70.1.9	Trident	645
70.1.10	BioKepler	646
70.1.11	Galaxy	646
70.1.12	Jupyter and IPython	646
70.1.13	Dryad – check content –	647
70.1.14	Naiad – check content –	647
70.1.15	Oozie	647
70.1.16	Tez	648
70.1.17	Google FlumeJava – check content –	648
70.1.18	Crunch	649
70.1.19	Cascading	649
70.1.20	Askalon	649
70.1.21	Scalding	650
70.1.22	e-Science Central – check content –	650
70.1.23	Azure Data Factory – check content –	650
70.1.24	Google Cloud Dataflow – check content –	650
70.1.25	NiFi (NSA)	651
70.1.26	Jitterbit	651

70.1.27	Talend	651
70.1.28	Pentaho	651
70.1.29	Apatar	652
70.1.30	Docker Compose	652
70.1.31	KeystoneML	652
70.2	Application and Analytics	653
70.2.1	Mahout	653
70.2.2	MLlib	653
70.2.3	MLbase	653
70.2.4	DataFu	653
70.2.5	R	654
70.2.6	pbdr	654
70.2.7	Bioconductor – check content –	654
70.2.8	ImageJ	654
70.2.9	OpenCV	655
70.2.10	Scalapack	655
70.2.11	PetSc	655
70.2.12	PLASMA MAGMA	655
70.2.13	Azure Machine Learning	656
70.2.14	Google Prediction API & Translation API	656
70.2.15	mlpy	656
70.2.16	scikit-learn	657
70.2.17	PyBrain	657
70.2.18	CompLearn	657
70.2.19	DAAL (Intel)	658
70.2.20	Caffe	658
70.2.21	Torch	658
70.2.22	Theano	658
70.2.23	DL4j	659
70.2.24	H2O	659
70.2.25	IBM Watson – check content –	659
70.2.26	Oracle PGX	660
70.2.27	GraphLab	660
70.2.28	GraphX	660
70.2.29	IBM System G	660
70.2.30	GraphBuilder (Intel)	661
70.2.31	TinkerPop	661
70.2.32	Parasol	661
70.2.33	Dream:Lab	661
70.2.34	Google Fusion Tables	662
70.2.35	CINET	662
70.2.36	NWB	662
70.2.37	Elasticsearch	663
70.2.38	Kibana	663
70.2.39	Logstash	663
70.2.40	Graylog	664
70.2.41	Splunk	664
70.2.42	Tableau	664
70.2.43	D3.js	664
70.2.44	three.js	665
70.2.45	Potree	665

70.2.46	DC.js	665
70.2.47	TensorFlow	665
70.2.48	CNTK	666
70.3	Application Hosting Frameworks	666
70.3.1	Google App Engine	666
70.3.2	AppScale	666
70.3.3	Red Hat OpenShift	667
70.3.4	Heroku	667
70.3.5	Aerobatic	667
70.3.6	AWS Elastic Beanstalk	667
70.3.7	Azure	668
70.3.8	Cloud Foundry	668
70.3.9	Pivotal	669
70.3.10	IBM BlueMix	669
70.3.11	(Ninefold)	669
70.3.12	Jelastic	669
70.3.13	Stackato	669
70.3.14	appfog	670
70.3.15	CloudBees	670
70.3.16	Engine Yard	670
70.3.17	(CloudControl)	671
70.3.18	dotCloud	671
70.3.19	Dokku	671
70.3.20	OSGi	671
70.3.21	HUBzero	671
70.3.22	OODT	671
70.3.23	Agave	671
70.3.24	Atmosphere	672
70.4	High level Programming	672
70.4.1	Kite	672
70.4.2	Hive	672
70.4.3	HCatalog	673
70.4.4	Tajo	673
70.4.5	Shark	673
70.4.6	Phoenix	674
70.4.7	Impala	674
70.4.8	MRQL	675
70.4.9	SAP HANA	675
70.4.10	HadoopDB	675
70.4.11	PolyBase	675
70.4.12	Pivotal HD/Hawq	676
70.4.13	Presto	676
70.4.14	Google Dremel	676
70.4.15	Google BigQuery	677
70.4.16	Amazon Redshift	677
70.4.17	Drill	677
70.4.18	Kyoto Cabinet	678
70.4.19	Pig	678
70.4.20	Sawzall	678
70.4.21	Google Cloud DataFlow	679
70.4.22	Summingbird	679

70.4.23	Lumberyard	679
70.5	Streams	680
70.5.1	Storm	680
70.5.2	S4	680
70.5.3	Samza	680
70.5.4	Granules	681
70.5.5	Neptune	681
70.5.6	Google MillWheel	681
70.5.7	Amazon Kinesis	681
70.5.8	LinkedIn	682
70.5.9	Twitter Heron – check content –	682
70.5.10	Databus	682
70.5.11	Facebook Puma/Ptail/Scribe/ODS	682
70.5.12	Azure Stream Analytics	683
70.5.13	Floe	683
70.5.14	Spark Streaming	683
70.5.15	Flink Streaming	684
70.5.16	DataTurbine	684
70.6	Basic Programming Model and Runtime, SPMD, MapReduce	684
70.6.1	Hadoop	684
70.6.2	Spark	684
70.6.3	Twister	685
70.6.4	MR-MPI	685
70.6.5	Stratosphere (Apache Flink)	685
70.6.6	Reef	685
70.6.7	Disco	686
70.6.8	Hama	686
70.6.9	Giraph	686
70.6.10	Pregel	686
70.6.11	Pegasus	686
70.6.12	Ligra	687
70.6.13	GraphChi	687
70.6.14	Galois	687
70.6.15	Medusa-GPU	687
70.6.16	MapGraph	688
70.6.17	Totem	688
70.7	Inter process communication Collectives	688
70.7.1	point-to-point	688
70.7.2	(a) publish-subscribe: MPI – check content –	688
70.7.3	(b) publish-subscribe: Big Data	688
70.7.4	HPX-5	689
70.7.5	Argo BEAST HPX-5 BEAST PULSAR	689
70.7.6	Harp	689
70.7.7	Netty	689
70.7.8	ZeroMQ	689
70.7.9	ActiveMQ	690
70.7.10	RabbitMQ	690
70.7.11	NaradaBrokering	691
70.7.12	QPid	691
70.7.13	Kafka	691

70.7.14	Kestrel	691
70.7.15	JMS	692
70.7.16	AMQP	692
70.7.17	Stomp	692
70.7.18	MQTT	693
70.7.19	Marionette Collective	693
70.7.20	Public Cloud: Amazon SNS	693
70.7.21	Lambda	694
70.7.22	Google Pub Sub	694
70.7.23	Azure Queues	694
70.7.24	Event Hubs	694
70.8	In-memory databases/caches	695
70.8.1	Gora (general object from NoSQL)	695
70.8.2	Memcached	695
70.8.3	Redis	695
70.8.4	LMDB (key value)	696
70.8.5	Hazelcast	696
70.8.6	Ehcache	696
70.8.7	Infinispan	697
70.8.8	VoltDB	697
70.8.9	H-Store	697
70.9	Object-relational mapping	698
70.9.1	Hibernate	698
70.9.2	OpenJPA	698
70.9.3	EclipseLink	698
70.9.4	DataNucleus	698
70.9.5	ODBC/JDBC	699
70.10	Extraction Tools	699
70.10.1	UIMA	699
70.10.2	Tika	700
70.11	SQL and SQL Services	700
70.11.1	Oracle	700
70.11.2	DB2	700
70.11.3	SQL Server	700
70.11.4	SQLite	701
70.11.5	MySQL	701
70.11.6	PostgreSQL	701
70.11.7	CUBRID	702
70.11.8	Galera Cluster	702
70.11.9	SciDB	702
70.11.10	Rasdaman	703
70.11.11	Apache Derby	703
70.11.12	Pivotal Greenplum	703
70.11.13	Google Cloud SQL	703
70.11.14	Azure SQL	704
70.11.15	Amazon RDS	704
70.11.16	Google F1	704
70.11.17	IBM dashDB	704
70.11.18	NTQL	705
70.11.19	BlinkDB	705

70.11.20	Spark SQL	705
70.12	NoSQL	705
70.12.1	Lucene	705
70.12.2	Solr	705
70.12.3	Solandra	705
70.12.4	Voldemort	706
70.12.5	Riak	706
70.12.6	ZHT	706
70.12.7	Berkeley DB	707
70.12.8	Kyoto/Tokyo Cabinet – check content –	707
70.12.9	Tycoon	707
70.12.10	Tyrant	708
70.12.11	MongoDB	708
70.12.12	Espresso	708
70.12.13	CouchDB – check content –	709
70.12.14	Couchbase Server	709
70.12.15	IBM Cloudant	710
70.12.16	Pivotal Gemfire	710
70.12.17	HBase	710
70.12.18	Google Bigtable	710
70.12.19	LevelDB	711
70.12.20	Megastore and Spanner	711
70.12.21	Accumulo	711
70.12.22	Cassandra	712
70.12.23	RYA	712
70.12.24	Sqrl	712
70.12.25	Neo4J	712
70.12.26	graphdb	712
70.12.27	Yarcdata – check content –	713
70.12.28	AllegroGraph	713
70.12.29	Blazegraph	713
70.12.30	Facebook Tao	714
70.12.31	Titan:db	714
70.12.32	Jena	714
70.12.33	Sesame	715
70.12.34	Public Cloud: Azure Table	715
70.12.35	Amazon Dynamo	715
70.12.36	Google DataStore	716
70.13	File management	716
70.13.1	iRODS	716
70.13.2	NetCDF	716
70.13.3	CDF	717
70.13.4	HDF	717
70.13.5	OPeNDAP	717
70.13.6	FITS	717
70.13.7	RCFile	718
70.13.8	ORC	718
70.13.9	Parquet	718
70.14	Data Transport	718
70.14.1	BitTorrent	718

70.14.2	HTTP	719
70.14.3	FTP	719
70.14.4	SSH	719
70.14.5	Globus Online (GridFTP)	719
70.14.6	Flume	720
70.14.7	Sqoop	720
70.14.8	Pivotal GPLOAD/GPFDIST	720
70.15	Cluster Resource Management	720
70.15.1	Mesos	720
70.15.2	Yarn	721
70.15.3	Helix	721
70.15.4	Llama	722
70.15.5	Google Omega	722
70.15.6	Facebook Corona	722
70.15.7	Celery	722
70.15.8	HTCondor	723
70.15.9	SGE	723
70.15.10	OpenPBS	723
70.15.11	Moab	724
70.15.12	Slurm	724
70.15.13	Torque	724
70.15.14	Globus Tools – check content –	724
70.15.15	Pilot Jobs	725
70.16	File systems	725
70.16.1	HDFS	725
70.16.2	Swift	725
70.16.3	Haystack	725
70.16.4	f4	726
70.16.5	Cinder	726
70.16.6	Ceph	726
70.16.7	FUSE	727
70.16.8	Gluster	727
70.16.9	Lustre	727
70.16.10	IBM Spectrum Scale, formerly GPFS	727
70.16.11	GFFS	728
70.16.12	Public Cloud: Amazon S3	728
70.16.13	Azure Blob	728
70.16.14	Google Cloud Storage	729
70.17	Interoperability	729
70.17.1	Cloudmesh	729
70.17.2	Libvirt	729
70.17.3	Libcloud	730
70.17.4	JClouds	730
70.17.5	TOSCA	731
70.17.6	OCCI	731
70.17.7	CDMI	731
70.17.8	Whirr	732
70.17.9	Saga	732
70.17.10	Genesis	732

70.18	DevOps	732
70.18.1	Docker (Machine, Swarm)	732
70.18.2	Puppet	733
70.18.3	Chef	733
70.18.4	Ansible	733
70.18.5	SaltStack	734
70.18.6	Boto	734
70.18.7	Cobbler	734
70.18.8	Xcat	734
70.18.9	Razor	735
70.18.10	Juju	735
70.18.11	Foreman	735
70.18.12	OpenStack Heat	735
70.18.13	Sahara	736
70.18.14	Rocks	736
70.18.15	Cisco Intelligent Automation for Cloud	736
70.18.16	Ubuntu MaaS	736
70.18.17	Facebook Tupperware	736
70.18.18	AWS OpsWorks	737
70.18.19	OpenStack Ironic	737
70.18.20	Google Kubernetes	737
70.18.21	Buildstep	738
70.18.22	Gitreceive	738
70.18.23	OpenTOSCA	738
70.18.24	Winery – check content –	738
70.18.25	CloudML	739
70.18.26	Blueprints – check content –	739
70.18.27	Terraform – check content –	739
70.18.28	DevOpSlang	739
70.18.29	Any2Api – check content –	740
70.19	IaaS Management from HPC to hypervisors	740
70.19.1	Xen	740
70.19.2	KVM	741
70.19.3	QEMU – check content –	741
70.19.4	Hyper-V – check content –	741
70.19.5	VirtualBox	741
70.19.6	OpenVZ – check content –	742
70.19.7	LXC – check content –	742
70.19.8	Linux-Vserver – check content –	742
70.19.9	OpenStack – check content –	743
70.19.10	OpenNebula – check content –	743
70.19.11	Eucalyptus	743
70.19.12	Nimbus – check content –	743
70.19.13	CloudStack – check content –	744
70.19.14	CoreOS – check content –	744
70.19.15	rkt – check content –	744
70.19.16	VMware ESXi	745
70.19.17	vSphere and vCloud – check content –	745
70.19.18	Amazon – check content –	745
70.19.19	Azure	746
70.19.20	Google and other public Clouds	746

70.19.21	Networking: Google Cloud DNS	746
70.19.22	Amazon Route 53	746
70.20	Cross-Cutting Functions	747
70.20.1	Monitoring	747
70.20.2	Security and Privacy	748
70.20.3	Distributed Coordination	750
70.21	Message and Data Protocols	751
70.21.1	MQTT	751
70.21.2	Avro	751
70.21.3	Thrift – check content –	751
70.21.4	Protobuf – check content –	751
70.22	Technologies To Be Integrated	751
70.22.1	Snort – check content –	751
70.22.2	Fiddler	752
70.22.3	Zeppelin – check content –	752
70.22.4	Open MPI – check content –	752
70.22.5	Apache Tomcat – check content –	752
70.22.6	Apache Beam – check content –	752
70.22.7	Cloudability – check content –	753
70.22.8	CUDA – check content –	753
70.22.9	Blaze – check content –	753
70.22.10	CDAP – check content –	753
70.22.11	Apache Arrow – check content –	754
70.22.12	OpenRefine	754
70.22.13	Apache OODT – check content –	754
70.22.14	Omid – check content –	755
70.22.15	Apache Ant	755
70.22.16	LXD – check content –	755
70.22.17	Wink – check content –	755
70.22.18	Apache Apex	756
70.22.19	Apache Knox – check content –	756
70.22.20	Apache Apex	756
70.22.21	Robot Operating System (ROS)	757
70.22.22	Apache Flex – check content –	757
70.22.23	Apache Ranger – check content –	757
70.22.24	Google Cloud Machine Learning	757
70.22.25	Karajan	758
70.23	Exercise	758

XX

Draft: IoT

71	Introduction	763
72	Hardware for IoT Projects	765
72.1	Raspberry Pi 3	765
72.2	ESP8266 Robot Car Kit	766
72.3	Sensor Kit	767

72.4	Fish Kit	767
72.5	Alternative components	767
72.5.1	Esp8266 Alternatives	767
72.5.2	Car Parts Alternatives	767
72.5.3	Simple sensors	767
72.5.4	Grove Sensors	768
72.6	Alternative Hardware and Sensors	768
72.6.1	Small Footprint Battery Power	768
73	Projects	769
74	ESP8266	771
74.0.1	Installation	771
74.0.2	Option B: setup from pip	773
74.0.3	Option C: A possible setup for Linux	774
74.0.4	Using cloudmesh robot	775
74.0.5	Resources	779
74.0.6	Alternative boards	780
74.0.7	Appendix	781
74.0.8	Installing ESP8266 USB drivers	781
75	Raspberry PI 3	783
75.1	Raspbery PI for IOT (Gregor)	783
75.2	Hardware	783
75.3	Installation	783
75.3.1	Erasing the SD Card	783
75.3.2	Installation of NOOBS	784
75.3.3	Installation of Dexter	784
75.4	Configure	784
75.4.1	Prepare OS	784
75.5	Update	784
75.5.1	Update to Python 3.6.1	785
75.6	change python version	785
75.7	install 3.6.1	786
75.8	install cloudmesh.pi	786
75.8.1	Install scientific Libraries	786
75.8.2	cloudmesh.pi (Jon)	786
75.8.3	Install VNC	787
75.9	Sensors (Jon)	787
75.9.1	Grove Sensors (Jon)	787
75.9.2	Non Grove Sensors (Jon)	787
75.10	Notes To integrates	787
75.10.1	Connecting	787
75.11	VLC on OSX	787
75.12	Camera on Pi	787
75.13	Streaming video	788

75.14	Linux Commandline	788
75.15	Enable SPI	788
75.16	RTIMULib2	788
75.17	Compile RTIMULib Apps	789
75.18	Camera	790
75.19	Lessons and Projects	790
75.20	OTHER TO BE INTEGRATED	791
75.20.1	PI emulator on Windows	791
75.20.2	Scratch	791
75.21	Web Server	791
76	Dexter	793
76.1	Creating an SD Card	793
76.1.1	OSX	793
76.1.2	Dexter Rasbian	793
76.1.3	Github	794
76.1.4	Cloning Grove Pi	794
76.1.5	Dexter Sample programs	794
77	GrovePi Modules	795
77.1	Introduction	795
77.2	LED	795
77.3	Buzzer	796
77.4	Relay	796
77.5	Light Sensor	797
77.6	Rotary Angle Sensor	797
77.7	Barometer	797
77.8	Distance Sensor	798
77.9	Temperature Sensor	798
77.10	Heartbeat Sensor	799
77.11	Joystick	799
77.12	LCD Screen	799
77.13	Moisture Sensor	799
77.14	Water Sensor	802
77.15	Sensors	802
77.15.1	Compass	802
78	VNC	803
78.1	Setting up VNC	803
78.1.1	Install VNC on OSX	803
78.1.2	Run VNC Viewer on OSX	804

79	Turtle Graphics	805
79.1	Demo	805
79.2	Program example	805
79.3	Shape	806
79.4	Links	806
79.5	Robot Dance Simulator	806
79.6	Scratch	806
79.7	MBlock	807
80	Tools	809
80.1	Markdown	810
80.2	Aquamacs	810
80.3	Bash	811
80.4	Arduino	811
80.5	OSX Terminal	811

XXI

—— Draft Chapters and Parts ——

XXII

Draft: Incoming Contributions

80.6	Python Apache Avro	817
80.6.1	Download, Unzip and Install	817
80.6.2	Defining a schema	817
80.6.3	Serializing	818
80.6.4	Deserializing	818
80.6.5	Resources	819
80.7	Creating a Virtual Machine on Google Compute Engine	819
80.7.1	Requirements	819
80.7.2	Automated Creation of a VM on Ubuntu 16.04	820
80.7.3	Create an Instance	820
80.7.4	MongoDB Tutorial	821
80.8	PI DHCP Server Tutorial	822
80.8.1	Introduction	822
80.8.2	Setting up the DHCP server	823
80.8.3	Configure fixed IP's for clients	826
80.8.4	Resources	827
80.9	VERSION A: Install Docker on a Raspberry Pi	827
80.10	VERSION B: Pi Docker	827
80.10.1	Preparing the SD card	827
80.10.2	Download Etcher here:	828
80.10.3	Enable SSH on the SD Card	828
80.10.4	Starting Pi	828
80.10.5	Docker Installation	828
80.10.6	Run apt-get update	828
80.10.7	Install Docker	828

80.10.8	Configure Docker	828
80.10.9	Enable Docker client	829
80.10.10	Test Docker	829
80.11	Rasberry Pi Hadoop Spark Cluster	829
80.11.1	Initial Configuration	829
80.11.2	Creating a new group and user	830
80.11.3	Generating the ssh keys (Perform these steps in master and all the workers)	830
80.11.4	Installing Hadoop	831
80.11.5	Setting the Environment Variables by modifying the bashrc file	831
80.11.6	Configure hadoop 2.7.1 (Perform this on the master and all the other workers)	832
80.11.7	Edit the XMI Files	832
80.11.8	Steps for Master	832
80.11.9	Edit XML files for the workers	835
80.11.10	Prepare the HDFS directory by executing the below commands	835
80.11.11	Booting Hadoop	835
80.11.12	Master - Worker Configuration	835
80.11.13	Configurtion in Worker servers	836
80.11.14	Prepare the HDFS directory by executing the below commands	837
80.11.15	Booting Hadoop	837
80.11.16	REBOOT and start services	838
80.11.17	TESTING THE CONFIGURATION OF MASTER_SLAVE	838
80.11.18	ISSUES TO BE ADDRESSED	838
80.12	Raspberry Pi Kubernetes Cluster	838
80.12.1	Note	838
80.12.2	Hardware	838
80.12.3	Configuration	839
80.13	Intructions for installing kubernetes	841
80.13.1	First install doker, disable swap, install kubeadm	841
80.13.2	For the nodes	841
80.13.3	Now some more explanations on the scripts	841
80.13.4	Steps for Setting up the Spark on the Rasberry Pi Cluster	842
80.14	Install Raspbian on an SD card using MacOS	843
80.14.1	Overview	843
80.14.2	Method1 - Without using NOOBS	843
80.14.3	Method2 - Using NOOBS	843
80.14.4	Resources	844
80.14.5	Create a Raspian SD-Card from a Ubuntu Machine	844
80.15	Adding a Public keys to a Raspbery Pi.	845
80.15.1	SD Card method	845
80.15.2	Network Connected Method	845
80.15.3	Copying the public key by hand	845
80.15.4	Travis CI	846
81	Contributed Tutorials	849
81.1	Open Stack	849
81.2	SSH	849
81.3	MapReduce	849

81.4	Pi	849
81.5	Aws	850
81.6	Other	850

XXIII **Draft: Operating System**

81.7	Ubuntu on a USB stick for OSX	853
81.7.1	Ubuntu on a USB stick for OSX via Command Line	853
81.7.2	Boot from the USB Stick	855
81.7.3	Ubuntu on a USB stick for OSX via GUI	855

XXIV **Draft: Pi**

81.8	Pi Cluster Related Information	861
------	---	-----

XXV **Draft: Big Data Algorithms**

82 **Technology Training - kNN and Clustering**

82.1	Recommender Systems - K-Nearest Neighbors	865
82.1.1	Python k'th Nearest Neighbor Algorithms	865
82.1.2	3D Visualization	866
82.1.3	Testing k'th Nearest Neighbor Algorithms	866
82.2	Clustering and heuristic methods	866
82.2.1	Kmeans Clustering	866
82.2.2	Clustering of Recommender System Example	866
82.2.3	Clustering of Recommender Example into more than 3 Clusters	866
82.2.4	Local Optima in Clustering	866
82.2.5	Clustering in General	867
82.2.6	Heuristics	867
82.3	Resources	867

83 **Technology for Big Data Applications and Analytics**

83.1	Technologypi: K-means	869
83.1.1	K-means in Python	870
83.1.2	Analysis of 4 Artificial Clusters	870
83.2	Technology: MapReduce	870
83.2.1	Introduction	870
83.2.2	Advanced Topics	870
83.3	Technology: Kmeans and MapReduce Parallelism	871
83.3.1	MapReduce Kmeans in Python	871
83.4	Technology: PageRank	871
83.4.1	Calculate PageRank from Web Linkage Matrix	871
83.4.2	Calculate PageRank of a Real Page	871

84	Technology Training - Plotviz	873
84.1	Using Plotviz Software for Displaying Point Distributions in 3D	873
84.1.1	Motivation and Introduction to use	874
84.1.2	Example of Use I: Cube and Structured Dataset	874
84.1.3	Example of Use II: Proteomics and Synchronized Rotation	874
84.1.4	Example of Use III: More Features and larger Proteomics Sample	874
84.1.5	Example of Use IV: Tools and Examples	874
84.1.6	Example of Use V: Final Examples	874
84.1.7	Resources	875

XXVI **Draft: Artificial Intelligence**

XXVI **Outdated: Cloud Computing**

85	OUTDATED: Cloud Computing Fundamentals	883
85.1	Retired lectures	883
85.2	Data Center Model	883
85.3	Data Intensive Sciences	883
85.4	IaaS, PaaS and SaaS	884
85.5	Challenges	884
86	Outdated: IaaS	885
86.1	Growth of Virtual Machines	885
86.2	Implementation Levels	885
86.3	Tools and Mechanisms	886
86.4	CPU, Memory & I/O Devices	886
86.5	Clusters and Resource Management	886
86.6	Data Center Automation	887
86.7	Clouds in the Workplace	887
86.8	Checklists and Challenges	887
86.9	Data Center Setup	887
86.10	Cultivating Clouds	888

XXVI **Outdated: Data Management**

87	Outdated: NoSQL	891
87.1	RDBMS vs. NoSQL	891
87.2	NoSQL Characteristics	891
87.3	BigTable	892
87.4	HBase	892
87.5	HBase Coding	892

87.6	Indexing Applications	892
87.7	Related Work	893
87.8	Indexamples	893
87.9	Indexing 101	893
87.10	Social Media Searches	893
87.11	Analysis Algorithms	894

XXIX **Outdated: SaaS**

88	Outdated: Search Engine	897
88.1	Google Components	897
88.2	Google Architecture	897
88.3	Google History	897

XXX **Outdated: MapReduce**

89	Outdated: MapReduce	901
89.1	Apache Data Analysis OpenStack	901
89.2	MapReduce	901
89.3	Hadoop Framework	902
89.4	Hadoop Tasks	902
89.5	Fault Tolerance	902
89.6	Hadoop WordCount on VMs	902
89.7	Programming on a Compute Cluster	903
89.8	How Hadoop Runs on a MapReduceJob	903
89.9	Literature Review	903
89.10	Introduction to BLAST	903
89.11	BLAST Parallelization	904
89.12	SIMD vs MIMD;SPMD vs MPMD	904
89.13	Data Locality	904
89.14	Optimal Data Locality	904
89.15	Task Granularity	905
89.16	Resource Utilization and Speculative Execution	905
90	Outdated: Iterative Map Reduce	907
90.1	Introduction to MapReduce	907
90.2	Google Search Engine 1	907
90.3	Google Search Engine 2	908
90.4	Hadoop PageRank	908
90.5	Discussions and ParallelThinking	908

90.6	Hadoop Extensions	908
90.7	Iterative MapReduce Models	909
90.8	Parallel Processes	909
90.9	Static and Variable Data	909
90.10	MapReduce Model Comparison	910
90.11	Twister K-means	910
90.12	Coding and Iterative Alternatives	910

XXXI Outdated: Internet of Things

91	Outdated: IoT	915
91.1	Everyday Data	915
91.2	Streaming the Data Ocean	915
91.3	Streams of Events	915
91.4	Faults & Frameworks	916
91.5	Spouts to Bolts	916

XXXI Class Projects

92	Projects	919
92.1	Class: E516	919
92.1.1	Scope of Project	919
92.1.2	Deliverables	919
92.1.3	Helpful Chapters	920
92.2	Class: E616	920
92.2.1	Scope of Project	920
92.2.2	Deliverables	921
92.2.3	Helpful Chapters	921
92.3	Class: i524	921
92.3.1	Scope of Project	921
92.3.2	Deliverables	922
92.3.3	Helpful Chapters	922
92.4	Class: E222	922
92.4.1	Deliverables	923
	Bibliography	925
	Index	967



Todo List

TODO: An example todo	64
TODO: TA: add hyperlinks	79
TODO: TA: add hyperlinks	88
TODO: TA: add hyperlinks	95
TODO: Tyler: include the links to Geoffreys lectures he presented, didn't see the slides presented on Thursday Feb 1,2018 about parallel computing, Amdahls Law I beleive was the first slide	95
TODO: Tyler:Describe how each student gets a unique assignment. Coordinate that assignment.	97
TODO: Tyler: please include links to the sections	98
TODO: Describe use of biber instead of bibtex	177
TODO: Tyler: include cite for jabref, bibtex, biber at appropriate location in this section. Add them to the bib file.	178
TODO: describe ssh config	217
TODO: describe port forwarding	217
TODO: provide us with screenshots of the windows.	218
TODO: lessons-ssh-generate-key	226
TODO: This section has to be redone as we use class specific clones and not the master	231
TODO: Tyler: Include image of the issues on hid-sample	233
TODO: TA: some of the python examples assume REPL, but its better to use a print statement instead as more general, please fix	303
TODO: contribute	315
TODO: contribute	316
TODO: contribute	316
TODO: Tutorial: Please contribute a XML python tutorial.	319
TODO: Students: beautiful soup contribute tutorial	320
TODO: Students: define conventional database access tutorial	321
TODO: Students: defineSQLite database access tutorial	321
TODO: image missing	371

TODO: citation	372
TODO: This example is not tested. Please provide feedback and improve	418
TODO: TO BE CONTRIBUTED BY THE STUDENTS OF THE CLASS AS HOMEWORK	421
TODO: TO BE CONTRIBUTED BY THE STUDENTS OF THE CLASS AS HOMEWORK. If you elect Windows 10. You could be using the online documentation provided by starting it on Windows, or running it in a docker container.	421
TODO: To be contributed by Averill Cate hid-sp18-???	447
TODO: TA: Docker on windows. Please improve and finalize the section	456
TODO: TAs: Move all section includes here for releasable container lectures	503
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	513
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	513
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	513
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	514
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	514
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	514
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	519
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	519
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	519
TODO: Gregor: Update Video Hadoop Urls to Youtube Urls	520
TODO: TA: complete harp section	538
TODO: TA: complete twister 2	540
TODO: Class: check the network hub if it is a good choice as the one we originally chose could not be brought in sufficient quantity.	576
TODO: Gregor: Add IoT section and PI configuration	577
TODO: Arnav sp18-503	580
TODO: Yue Guo sp18-508, Min Chen sp18-405	580
TODO: Keerthi sp18-602	580
TODO: Yue sp18-508	580
TODO: Keerthi sp18-602	580
TODO: Arnav sp18-503	581
TODO: Yue Guo sp18-508, Min Chen sp18-405	581
TODO: Keerthi sp18-602	581
TODO: Goutham sp18-401	581
TODO: Priyadarshini sp18-421, Ankita sp18-502	581
TODO: Surya sp18-418	581
TODO: Priyadarshini sp18-421, Ankita sp18-502	582
TODO: Tim sp18-526, Juliano sp18-601	585
TODO: Hao Tian sp18-524, Lin Qingyun sp18-515	587
TODO: Karan Kotabagi sp18-412, Manoj, Min sp18-405, Ramya sp18-406, Karan Kamatagi sp18-410	589
TODO: These resources have not all been checked to see if they still exist this is curretnly in progress	601
TODO: These two videos need to be uploaded to youtube	607
TODO: These resources have not all been checked to see if they still exist this is curretnly in progress	608
TODO: These resources have not all been checked to see if they still exist this is curretnly in progress	615
TODO: integrate physics-references.bib	621

TODO: These resources have not all been checked to see if they still exist this is currently in progress	627
TODO: These resources have not all been checked to see if they still exist this is currently in progress	631
TODO: on box but not available, move to google drive	639
TODO: slides are on box and not google drive Aspects of Big Data Applications,Slides https://iu.box.com/s/atgkxucop1lzftkunj8a12fe74x65na6	640
TODO: slides are on box and not google drive Big Data Applications and Generalizing their Structure,Slides https://iu.box.com/s/01dndtucmynekgehur00vktfgcpggc1r	640
TODO: Min, Bertholt: could some of the actions not be done when we burn the SD Cards. E.g. I wonder if i plugin an SDcard can we not set up all the file actions. Also even when we are locally on the machine, as this is the first install, could we not just copy or cat the information into the right files. Maybe we can write a cmd5 command <code>cms pi dhcp ...</code> to make this easy?	822
TODO: hid-sp18-421, hid-sp18-502, Is it possible to do this while plugging in the SD card and add the key there. The provided solution is not scalable.	845
TODO: What if there is already a public key on the PI? Add additional instructions in another section.	845
TODO: This step is unclear. It is not explained what connect your raspberry pi means	845
TODO: Arnav: the example is not linked. So we need to make sure that the example is also copied. I am not sure what the best way of doing this is, maybe we need a new example directory under the user cloudmesh such as cloudmesh/travis-example. In any case link to Repo is missing	847
TODO: Help needed: we need an extension to showcase how to use docker	848
TODO: These resources have not all been checked to see if they still exist this is currently in progress	867



Preface

1	Introduction	59
1.1	Citation	
1.2	Authors	
1.3	About	
1.4	Other Results	
1.5	Contributing	
1.6	Conventions	
1.7	Exercises	
2	Updates	67



1. Introduction

This document is the first one in a series of documents that are providing a comprehensive overview of Clouds, Big Data, and their technologies and applications. Material from these volumes are used selectively as part of this class.

Indiana University

They are developed within the Intelligent Systems Engineering department at Indiana University. This courses are for students who are interested in any component of the Masters or Ph.D. in Intelligent Systems Engineering. It is an advanced elective class.

The Series includes:

1. (This document) **Handbook of Clouds and Big Data**, Gregor von Laszewski, Geoffrey C. Fox, and Judy Qiu, Fall 2017, <http://cyberaide.org/papers/vonLaszewski-bigdata.pdf>
2. **Use Cases in Big Data Software and Analytics Vol. 1**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v1.pdf>
3. **Use Cases in Big Data Software and Analytics Vol. 2**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v2.pdf>
4. **Use Cases in Big Data Software and Analytics Vol. 3**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/vonLaszewski-projects-v3>
5. (Draft) **Big Data Software Vol 1.**, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper1/proceedings.pdf>
6. (Draft) **Big Data Software Vol 2.**, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper2/proceedings.pdf>
7. (Draft) **Big Data Projects**, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/project/projects.pdf>
8. (Draft) **Vol 7**, Gregor von Laszewski, Spring 2018, <http://cyberaide.org/papers/vonLaszewski-cloud-vol-7.pdf>
9. (Draft) **Vol 8**, Gregor von Laszewski, Spring 2018, <http://cyberaide.org/papers/>

[vonLaszewski-cloud-vol-8.pdf](#)

Our plan for this semester also includes the reorganization of some of the volumes while sorting the contributions into chapters with papers from the same topic.

1.1 Citation

The citation for this document is

Gregor von Laszewski, Geoffrey C. Fox, and Judy Qiu, Handbook of Clouds and Big Data -- Theory and Practice, Indiana University, Jan., 2018 Smith Research Center, Bloomington, IN 47408 <http://cyberaide.org/papers/vonLaszewski-bigdata.pdf>

The bibtex entry for this document is

```
@TechReport{las17handbook,
  author =      {Gregor von Laszewski and Geoffrey C. Fox and Judy Qiu},
  title =       {Handbook of Clouds and Big Data -- Theory and Practice},
  institution = {Indiana University},
  year =        {2018},
  OPTtype =     {Draft},
  address =     {Smith Research Center, Bloomington, IN 47408},
  month =       jan,
  url={https://tinyurl.com/cloudmesh/vonLaszewski-bigdata.pdf}
}
```

1.2 Authors

Gregor von Laszewski Gregor von Laszewski is an Assistant Director Digital Science Center (DSC) in the School of Informatics and Computing and Engineering at Indiana University. He holds also a position as Adjunct Professor in the Intelligent Systems Engineering Department. Previously he held Adjunct Professor positions at the Computer Science Department at Indiana University and University of North Texas. He received a Ph.D. from Syracuse University in computer science.

At IU He served as the architect of the FutureGrid project. His current interest and projects include cloud computing, big data, and scientific impact metrics, and edge computing. He initiated the Cloudmesh project which is a toolkit to enable cloud computing across various Cloud and Container IaaS such as OpenStack, AWS, Azure, docker, docker swarm, and kubernetes.

Geoffrey C. Fox Fox received a Ph.D. in Theoretical Physics from Cambridge University and is now distinguished professor of Informatics and Computing, and Physics at Indiana University where he is director of the Digital Science Center, Chair of Department of Intelligent Systems Engineering and Director of the Data Science program at the School of Informatics, Computing, and Engineering.

He currently works in applying computer science from infrastructure to analytics in Biology, Pathology, Sensor Clouds, Earthquake and Ice-sheet Science, Image processing, Deep Learning, Manufacturing, Network Science and Particle Physics. The infrastructure work is built around Software Defined Systems on Clouds and Clusters. The analytics focuses on scalable parallelism.

Judy Qiu is an Associate Professor in the School of Informatics and Computing at Indiana University. Her research interests focus on data-intensive computing at the intersection of cloud and multicore technologies, with an emphasis on life science applications using

MapReduce as well as traditional parallel and distributed computing approaches. Her contributions are focused on Hadoop and Introduction into Cloud Computing.

1.3 About

The material in this document is covering material that is or has been used in the following classes at Indiana University.

- Undergraduate: E222 Intelligent systems II
- Graduate: E516 Introduction to Cloud Computing
- Graduate: E616 Advanced Cloud Computing
- Graduate: E534 Big Data Applications
- Graduate: I524 Big Data Applications and Open Source Software
- Graduate: I523 Big Data Applications and Analytics

The format has been influenced by experiences gained from teaching I523 and I524 by Gregor von Laszewski. The collection of this material is updated continuously and new versions will be made available throughout the semester. A timestamp on the front page indicates the last time the document was updated.

1.4 Other Results

Besides the documents posted in Section 1 a small set of additional results exist in form of a video presentation. However, these results are older and we are no longer using such presentations. To guarantee completeness, we include them however in this section

Student Work 1 (8:48) 

Student Work 1 (Page 7) 

Student Work 1 (pptx) (Page 7) 

Student Work 2 (10:03) 

Student Work 2 (Page 12) 

Student Work 2 (pptx) (Page 12) 

1.5 Contributing

We encourage students of the classes to contribute to this document or other volumes, provide corrections, and additions. This document is managed on `github.com` at

- <https://github.com/cloudmesh/book/>

1.5.1 Class Contributions

The current release version is held in the master branch. Development versions will be held under a number of branches:

e222 Branch with contributions from students of the e222 class. Merges to and from the *latex* branch will be conducted on a daily bases by TA's.

- e516** Branch with contributions from students of the e616 class. Merges to and from the *latex* branch will be conducted on a daily bases by TA's.
- e616** Branch with contributions from students of the e616 class. Merges to and from the *latex* branch will be conducted on a daily bases by TA's.
- dev** Branch managed by Gregor and the TA's
- master** Branch that contains the current released version. This version is updated once a week from the branch *latex*.

Contributions are to be conducted as pull requests. It is important to keep the pull requests small and on a section or even paragraph base. This helps avoiding conflicts at time of checkin and is a common practice in large communities. It is not a good practice to work for weeks on improvements and than issue the pull request. For sure things will have changed and it will take you a long time to catch up.

The original document is based on selected material published and edited by Gregor von Laszewski at the following Web page

- <https://cloudmesh.github.io/classes/>

Based on feedback from students the desire to have the material in book format available was raised and we have implemented it.

1.5.2 Fixing Typos

The Github graphical interface makes it seamless to fix typos, which is an additional way to contribute and may be one of the most important and overlooked aspects of contributing. A short video shot by an E516 student clearly showcase how to fix a typo once you have discovered one.

Fix Typo Video (01:18) 

1.5.3 Community Contributions

It is easy to contribute to this document and we invite everyone to improve the material. To do so you need to fork the repository from

- <https://github.com/cloudmesh/book/>

and clone it. Make sure you check out the *dev* branch. Then you can modify information in the different files or add new sections. It is important that you make changes based on sections and than for them create a new pull request. This simplifies the review process. We will typically want only one file to be changed. Aslo before you issue your pull request make sure that no one else has already made changes. In that case we ask you to integrate them into your document.

1.5.4 Contributors

We like to acknowledge the following contributors that helped on this document. Please notify us with your name and a brief commend on what you contributed:

Descriptions provided in Section 70 were contributed by the following people that are either listed by full name or their github.com id:

Abhijit Thakre, Abhishek Gupta, Abhishek Naik, Ajit Balaga, Anurag Kumar Jain, Avadhoot Agasti, Badi' Abdul-Wahid, Cmbays, DIKSHA, Dimitar Nikolov, Govind, Govind Mishra, Grace Li, Gregor von Laszewski, Harshit Krishnakumar, Hyungro

Lee, Jerome Mitchell, Jimmy Ardiansyah, Jon, Jon Montgomery, Jordan Simmons, Juliette Zerick, Karthik, Kumar Satyam, Mark McCombe, Matthew Lawson, Methkupalli Vasanth, Miao Jiang, Miao Zhang, Milind Suryawanshi, MilindSuryawanshi, Nandita Sathe, Naveen, Niteesh01, Piyush Rai, Piyush Shinde, Prashanth, Pratik Jain, Rahul Raghatate, Rahul Singh, Ribka Rufael, Ronak Parekh, Saber Sheybani, Sabyasachi Roy Choudhury, Sagar Vora, Sahiti Korrapati, Scott McClary, Sean Shiverick, SilviaKarim14, Sivaprasad Sushmita, Snehal Chemburkar, Sowmya Ravi, Srikanth Ramanam, Sunanda Unni, SushmitaSivaprasad, Tony Liu, Vasanth Methkupalli, Veera Marni, Vibhatha Abeykoon, Vibhatha Lakmal Abeykoon, Vishwanath Kodre, William H Knapp III, acaastrob, ak.15, alyez, anveling, argetlam115, athakre, bhavesh37, cacoulte, cglmoocs, elenadesigner, eunosm3, harkrish1, jemitchell, justbbusy, jzerick, kartanba, karthick, karthick venkatesan, karthik-anba, kpvenkat, ksrivatsav, Imundia, miaozhan, michaelsmith1983, mmccombe, nsathe, piyurai, pratik11jain, ronak1182, sabyasachi087, shah0112, sriramsitharaman, suunni, tifabi, tonythomascn, vasanth, vibhatha, vkodre, vlabeyko, xl41, yatinsharma7, Min Chen

1.5.5 Section Contributors

Chameleon Cloud. *Chameleon Team and Gregor von Laszewski.* Chapter 20 focussing on Chameleon Cloud was copied with permission from the chameleoncloud.org website on Jan 1st, 2018 and represents the contribution of the Chameleon project team, developed under the NSF grant 1743358, Collaborative Research: Chameleon: A Large-Scale, Reconfigurable Experimental Environment for Cloud Research. Some content has been modified and added by Gregor von Laszewski to specifically target classes and education material targeted for Indiana University.

MQTT. *Arnav and Gregor von Laszewski.* Chapter 35

Dockerhub. *Min Chen and Gregor von Laszewski.* Section 44.1

1.6 Conventions

1.6.1 Videos

Videos to the class are referred to with embedded links into the PDF document as follows:

[Test Video \(25:36\)](#) 

An index will also be available in the index page that lists on which page the video has been added.

1.6.2 Slides

Sides

[Test slides \(10\)](#) 

1.6.3 Images

The video icon was copied from <http://www.freeiconspng.com/img/8039>.

1.6.4 URLs

The online version of this document contains a significant number of links. The links are either embedded or are directly visible. The color of the links is blue.

Direct URL: This is an example for a <https://github.com/cloudmesh/book/>

Embedded URL: This is an example for an embedded URL that points to the [source on github](#)

1.6.5 Variables

In some cases we need to communicate in programs and scripts the use of usernames or IP addresses for compute resources. These are indicated in brackets such as

```
1 <use your username>
2 <use your ip address>
```

In these cases you need to replace the text including the brackets with the desired information. In case they appear in shell scripts you could also set them in the shell with

```
1 export USERNAME=<use your username>
2 export IP=<use your ip address>
```

Make sure that you do not overwrite other environment variables, so its a good practice to check if they are not already set while using

```
1 echo $USERNAME
2 echo $IP
```

beforehand.

1.6.6 Boxes

Note

Notes are written in blue boxes and indicate a clarification or some important information that you do not want to overlook.

Warning

Warnings are written in red boxes and indicate a piece of information that you must not ignore.

Indiana University

Red boxes with Indiana University are information that relate to students that use this material as part of the courses offered at Indiana University.

To dos are highlighted in boxes with the keyword TODO. The offer opportunities for student to gather points for the discussion grade.

TODO: An example todo

1.6.7 Copy and Paste

First, we do not recommend to just copy and paste ever without reading the content. We recommend that you first read the entire section or sometimes even the chapter. Only after you understand what the command does, use it.

Second, to make it simple the handbook contains now a hyperlink to the LaTeX source of a page if you look at the lower right of the page itself. You can click on it and will be redirected to github. You can than locate the section that defines the listing and copy from there. This helps in cases you are unsure of spaces or minus signs or other characters including line breaks.

Third, It is natural that some issues may exist, but be reminded that the examples we have in the handbook have actually been tried before. Spelling errors naturally can occur, and we would be grateful in case you see an error help us fixing it. This is especially the case when it comes to version numbers. Often you may want to visit the original source and check if the recommended version numbers listed in the handbook re still up to date. This naturally has also the consequence that if a new version is available that you must double check the rest of the content in order not to overlook updates that have been provided by the developers.

1.6.8 Github and Travis and Pull Requests

The book is managed in github at the following location

<https://github.com/cloudmesh/book/>

All checkins are verified by travis

<https://travis-ci.org/cloudmesh/book>

YOu can improve any thing in the book by creating **small** pull requests. Do not spend time to do large pull requests, limit them to a single file or in some cases to a paragraph. Small pull requests are much easier to review and integrate.

1.6.9 Github Issues

In case you do not want o create your own pull request, but like to report an isse, please use not only the section number to report the issue, but include the URL of the file to be changed which is provided in the bottom of the PDF version, describe how it is now, and how it should be changed.

1.7 Exercises

Exercise 1.1 Inspect the PDF documents produced by previous classes. Note the differences between technology and application reviews and projects. ■

Exercise 1.2 Contribute to this document while finding a single spelling error. Before you do the pull request, make sure the document compiles. ■



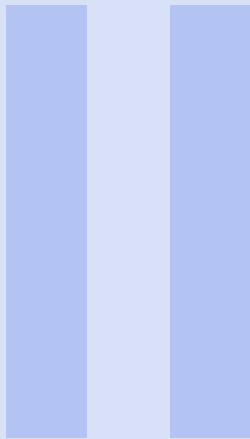
2. Updates

- Apr 06, 2018 Section [47.6](#) spark streaming lectures.
- Apr 06, 2018 Section [45](#) added virtual machines lectures.
- Apr 04, 2018: Section [49.2](#) added the Twister Installation added.
- Apr 06, 2018 [41.5](#) added about Hadoop 3.0.1 using Docker containers.
- Apr 06, 2018 [45](#) added virtual machines lectures.
- Apr 04, 2018: Section [49.2](#) added the Twister Installation added.
- April 6, 2018: Section [45](#) added with lecture slides and lab materials.
- April 6, 2018: Part ?? Section for E222 with introduction to k-nn with examples. Lecture notes with unsupervised learning, AI lab.
- Ma 29, 2018: Part [XXII](#) that contains incoming tutorials from the class was added. The students that have already committed their tutorials in `book/tutorial` must use that directory from now on and not submit or copy the tutorials under their hid. This is for now valid for all residential students. For online students we have for now just provided a list in Chapter [81](#). However best would be if those students also start adding the tutorials via a pull request and integrate them into the handbook. Some tutorials are obviously in draft form, so we like you to complete them. You may have noticed in some tutorials we have done significant modification. To simplify integration into the handbook we like you to provide the tutorial in md. Make sure to place images in an images directory, and bibtex in a bib file next to your tutorial. If unclear ask.
- Mar 27, 2018: Section [44.1](#) about docker hub has been added and updated
- Mar 26, 2018: Section [50](#) was updated and the first lectures about NoSQL for e516 were added. All subsections that are marked in the subsection title as *Draft*: are not yet released.
- Mar 26, 2018: Section [51](#) about NoSQL was updated
- Feb 27 - March 24, 2018: Although many of the Project ideas have been discussed in online meetings we summarized the discussions in general form in Section [8.16](#). We also point out again that a mandatory oral meeting needs to take place to define your project. Please make sure you update the google docs document.

- Mar 23, 2018: Added graph representation of topics covered for I524, E616 in Section [5.9.1](#) and E516 in Section [6.9.1](#).
- Mar 22, 2018: Added Hadoop Section [46](#).
- Mar 22, 2018: Added Spark Section [47](#).
- Mar 11, 2018: Updating Section so that openrc file can be downloaded and modified without GUI. Changing the path to `.cloudmesh`.
- Mar 10, 2018: Adding Section [62](#) for the i516 class.
- Mar 10, 2018: Moving all Application lectures for i524 and e616 into the Part [XVIII](#).
- Mar 7, 2018: Section [15.6](#), was added with a small introductory overview about Makefiles.
- Feb 26, 2018: Section [??](#) was added, describing how to fork a github repository.
- Feb 24, 2018: A few updates in Section [12.1](#) were made to indicate the prefix of bibtex labels with the hid. Some other minor issues were done to avoid copy and paste of content.
- Feb 22, 2018: Section [8.2](#) explains that we teach cloud computing and as part of cloud computing we want you to use a professional task management system such as github issues. It is an assignment of this class that you manage your assignments through github issues, a cloud service used by millions of professionals.
- Feb 21, 2018: Section [39.3](#) was added, describing how to deploy a flask REST application using docker.
- Feb 15, 2018: added information about chktex in Section [11.6.4](#) and lacheck in Section [11.6.3](#).
- Feb 13, 2018: A Chapter [35](#) about was added. A better introductory example is needed, If you like to contribute, please send me your contribution. Lets coordinate before you start.
- Mar 22, 2018: Section [47](#) was added, describing Spark.
- Mar 22, 2018: Section [47.5](#) was added, describing how install Spark locally.
- Mar 22, 2018: Section [46](#) was added, describing Hadoop.
- Mar 22, 2018: Section [46.7](#) was added, describing how to install Hadoop locally..
- Feb 11, 2018: A number of videos have been released.

<input type="checkbox"/> ✓ 33.1.1. Introduction - Part A	Released, Feb 11, 2018	400
<input type="checkbox"/> ✓ 33.1.2. Introduction - Part B - Defining Clouds I	Released, Feb 11, 2018	400
<input type="checkbox"/> ✓ 33.1.3. Introduction - Part C - Defining Clouds II	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.4. Introduction - Part D - Defining Clouds III	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.5. Introduction - Part E - Virtualization	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.6. Introduction - Part F - Technology Hypecycle I	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.7. Introduction - Part G - Technology Hypecycle II	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.8. Introduction - Part H - IaaS I	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.9. Introduction - Part I - IaaS II	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.10. Introduction - Part J - Cloud Software	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.11. Introduction - Part K - Applications I	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.13. Introduction - Part M - Applications III	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.14. Introduction - Part N - Parallelism	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.15. Introduction - Part O - Storage	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.16. Introduction - Part P - HPC in the Cloud	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.17. Introduction - Part Q - Analytics and Simulation	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.18. Introduction - Part R - Jobs	Released, Feb 11, 2018	404
<input type="checkbox"/> ✓ 33.1.19. Introduction - Part S - The Future	Released, Feb 11, 2018	404
<input type="checkbox"/> ✓ 33.1.20. Introduction - Part T - Security	Released, Feb 11, 2018	404
<input type="checkbox"/> ✓ 33.1.21. Introduction - Part U - Fault Tolerance	Released, Feb 11, 2018	404
- Feb 11, 2018: Added Video about Containers in Section [37](#).

-
- | | | |
|--|----------------------|---------------------|
| <input type="checkbox"/> ✓ 37.1. Motivation - Microservices | release Feb 11, 2018 | 451 |
| <input type="checkbox"/> ✓ 37.2. Motivation - Serverless Computing | release Feb 11, 2018 | 451 |
| <input type="checkbox"/> ✓ 37.3. Docker | release Feb 11, 2018 | 451 |
| <input type="checkbox"/> ✓ 37.4. Docker and Kubernetes | release Feb 11, 2018 | 452 |
- Feb 11,2018: Added description for swagger toolbox to generate models from json schemas. See Section [34.10.1](#).
 - Feb 10, 2018: Added outline for user community contributed sections
 - Feb 9, 2018: Add IoT draft sections for esp8266 and Raspberry PI sensors and projects
 - Feb 8, 2018: Improve laszewski.cls class and Makefile in order to simplify using the template in sharelatex and making it look like an ordinary book class
 - Feb 8, 2018: Switch from dev to master branch to support editing the handbook in sharelatex
 - Feb 5, 2018: First draft of container section
 - Feb 5, 2018: Add 2nd short table of contents
 - Jan 31, 2018: Add Section [17.19](#) about github issues.
 - Jan 27, 2018: The section HATEOAS was added, see Section [34.3.11](#) as well as some others.
 - Jan 25, 2018: added the draft Section [55](#) describing the form factor for several Raspberry PI Clusters
 - Jan 24, 2018: added Rest Chapters [34](#) and [34.11](#)
 - Jan 21, 2018: added the assignment Chapter [7](#).
 - Jan 21, 2018: added proposed weekly schedule for E516, I524 and E616
 - Jan 13, 2018: added course management video to the syllabus. E222 not yet included. TA's will work with Geoffrey on that.
 - Jan 11, 2018: changed the handbook link as github has bandwidth limits
 - Jan 10, 2018: Added facilities statement for FutureSystems (see Section [19.7](#)).
 - Jan 10, 2018: Wait list augmentation, based on new School Policy. You need to be registered to get credit (see Section [8.11](#)). Does probably not apply for anyone in this class.



Syllabus

3	Calendar	75
4	E516: Introduction to Cloud Computing	77
4.1	Course Description	
4.2	Course Prerequisites	
4.3	Registration Information	
4.4	Teaching and Learning Methods	
4.5	Covered Topics	
4.6	Student learning outcomes	
4.7	Grading	
4.8	Representative bibliography	
4.9	Lectures and Lecture Material	
5	E616/I524: Advanced Cloud Computing	85
5.1	Course Description	
5.2	Course pre-requisites	
5.3	Course Registration	
5.4	Teaching and learning methods	
5.5	Covered Topics	
5.6	Student learning outcomes	
5.7	Grading	
5.8	Representative bibliography	
5.9	Lectures and Lecture Material	
5.10	Containers	
6	E222: Intelligent Systems Engineering II	93
6.1	Course Description	
6.2	Course pre-requisites	
6.3	Course Registration	
6.4	Teaching and learning methods	
6.5	Covered Topics	
6.6	Student learning outcomes	
6.7	Grading	
6.8	Representative bibliography	
6.9	Lectures and Lecture Material	
7	Assignments	101
7.1	Assignments E222	
7.2	Assignments E516, I524, E616	



3. Calendar

- Week 1** Jan 12, 2018
- Week 2** Jan 15
- Week 3** Jan 22
- Week 4** Jan 29
- Week 5** Feb 5
- Week 6** Feb 12
- Week 7** Feb 19
- Week 8** Feb 26
- Week 9** Mar 5
- Week 10** Mar 11 - Mar 18 Spring break starts
- Week 11** Mar 19
- Week 12** Mar 26
- Week 13** Apr 2
- Week 14** Apr 9
- Week 15** Apr 16
- Week 16** Apr 23
- Week 17** Apr 30
- Week 18** May 4 (Fri) - Semester ends
- Week 19**



4. E516: Introduction to Cloud Computing

We present the syllabus for the introduction to Cloud Computing course taught at Indiana University that uses in part the material presented in this document. The information includes the section or chapter number, the name of the chapter or section, and the timeframe when it is recommended to work through the material and the page number where to find the material in this document. Please note that we will update the document throughout the semester thus, page numbers will change.

4.1 Course Description

This course describes the emerging cloud and big data technologies within the world of distributed intelligent systems where each system component has internal, and external sources of intelligence that are subject to collective control. Examples are given from a wide variety of applications. A project will allow students to dive into practical issues after they have obtained a theoretical background.

4.2 Course Prerequisites

Python will be used as a Programming Language. In some cases Java may also be useful however its use in class will be marginal. The class will be using Linux commandline tools. Prior knowledge of Linux is of advantage but not required. Students are expected to have access to a computer on which they can execute Linux easily. As the OS requirements have recently increased we recommend a computer with 8GB main memory and the ability to run virtualbox and/or containers. If it turns out your machine is not capable enough we attempt to provide access to IU linux machines.

4.3 Registration Information

ENGR-E 516 ENGINEERING CLOUD COMPUTING (3 CR)
33699 RSTR 11:15A-12:30P F I2 150 Von Laszewski G

Above class taught online
 Above class open to graduates only
 Discussion (DIS)

ENGR-E 516 ENGINEERING CLOUD COMPUTING (3 CR)
 33909 RSTR ARR ARR WB WEB Von Laszewski G
 This is a 100% online class taught by IU Bloomington. No
 on-campus class meetings are required. A distance education
 fee may apply; check your campus bursar website for more
 information
 Above class open to graduates only

4.4 Teaching and Learning Methods

- Lectures
- Assignments including specific lab activities
- Final project

4.5 Covered Topics

The class will cover a number of topics as part of tracks that are executed in parallel throughout the class. Although we assume that at a graduate course level the communication track has already been covered elsewhere, we made sure we also include it here in case you did not yet have such experiences.

- **Week 1: Course Overview:** Overview and status of cloud
- **Week 2-4: Introduction:** Overview and status of cloud computing. This includes the creation of REST services for Big Data
- **Week 6: Cloud Computing:** Basics of Cloud Computing
- **Week 7: Containers:** Basics of Containers
- **Week 8: Virtual Machines:** Introduction to OpenStack
- **Week 9: Cloud 3.0:** Microservices, Events, Functions
- **Week 10-13: Hadoop:** Introduction to Hadoop
- **Week 13-15: Project:** Delivery of a reproducible substantial student project

4.6 Student learning outcomes

When students complete this course, they should be able to:

- Have an elementary understanding of issues involved in designing and Software Defined Systems.
- Understand the concepts of Cloud Computing
- Gain hands-on laboratory experience with several examples.
- Apply knowledge of mathematics, science, and engineering
- Have advanced skills in teamwork with peers.

4.7 Grading

Grade Item	Percentage
Assignments	40%
Final Project	50%
Participation	10%

4.8 Representative bibliography

1. Cloud Computing for Science and Engineering By Ian Foster and Dennis B. Gannon
 - <https://mitpress.mit.edu/books/cloud-computing-science-and-engineering>
2. Ansible Tutorials
3. <http://bigdataopensourceprojects.soic.indiana.edu/>
4. The backdrop for course is the 350 software subsystems illustrated at <http://hpc-abds.org/kaleidoscope/>
5. http://cloudmesh.github.io/introduction_to_cloud_computing/
6. There are a huge number of web resources
7. (This document) **Handbook of Clouds and Big Data**, Gregor von Laszewski, Geoffrey C. Fox, and Judy Qiu, Fall 2017, <https://tinyurl.com/vonLaszewski-handbook>
8. **Use Cases in Big Data Software and Analytics Vol. 1**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v1.pdf>
9. **Use Cases in Big Data Software and Analytics Vol. 2**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v2.pdf>
10. **Use Cases in Big Data Software and Analytics Vol. 3**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/vonLaszewski-projects-v3>
11. (Draft) **Big Data Software Vol 1**, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper1/proceedings.pdf>
12. (Draft) **Big Data Software Vol 2**, Gregor von Laszewski, Spring 2017,
 - <https://github.com/cloudmesh/sp17-i524/blob/master/paper2/proceedings.pdf>
13. (Draft) **Big Data Projects**, Gregor von Laszewski, Spring 2017,
 - <https://github.com/cloudmesh/sp17-i524/blob/master/project/projects.pdf>


4.9 Lectures and Lecture Material

Warning

This section will change and be updated throughout the semester

Indiana University

To start the class at IU, we require you to watch the following video:

[Class Management - Overview \(39:54\)](#) 

[Teaching Cloud and Big Data - Overview \(39:54\)](#) 

4.9.1 Class Graph E516

We present the Class outline als in a graph representation in Figure 4.1. Hyperlinks to the specific sections are included

TODO: TA: add hyperlinks

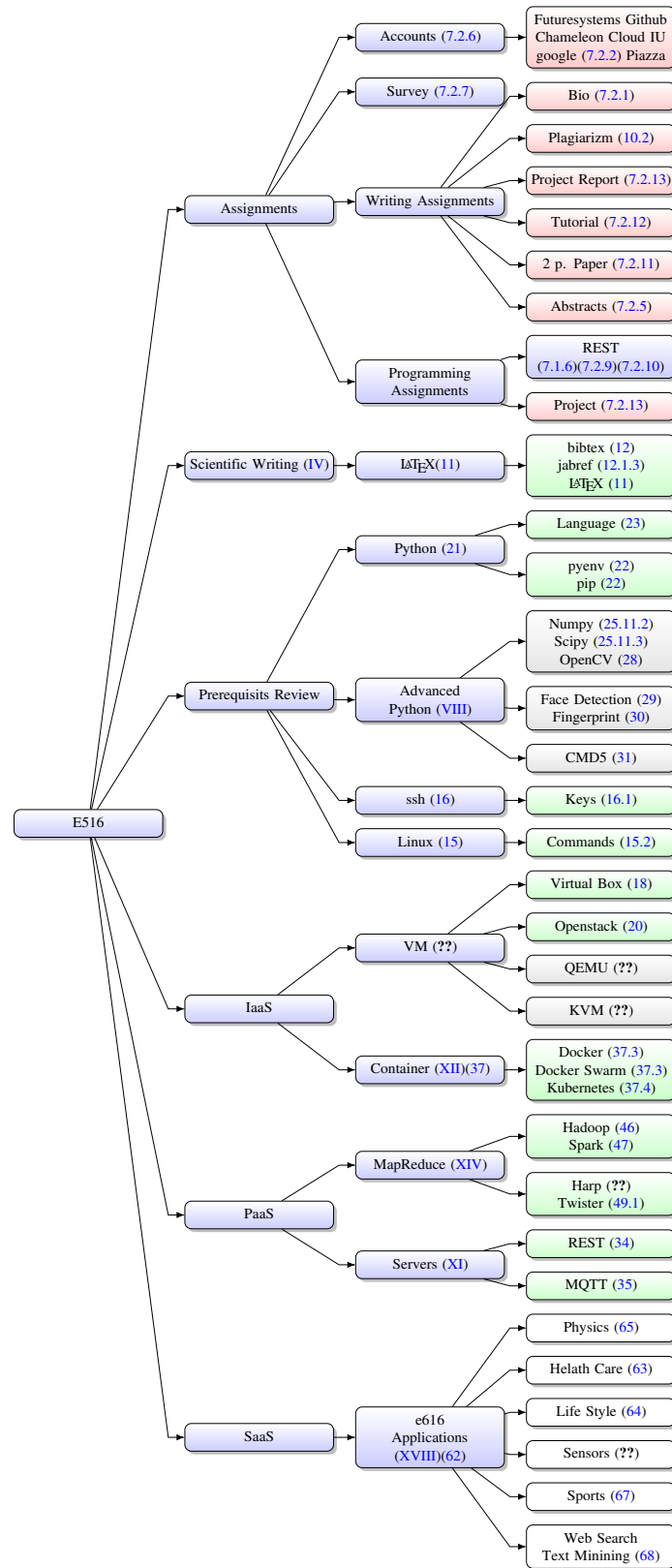


Figure 4.1: E516 class structure.

4.9.2 Assignments

All assignments are summarized in the Section 7.2

The current released assignments include:

<input type="checkbox"/> ✓ 7.2.3. Big Data Collaboration	Due: Jan 29, 2018	104
<input type="checkbox"/> ✓ 7.2.4. New Technology List	Due: Jan 29, 2018	105
<input type="checkbox"/> ✓ 7.2.5. New Technology Abstract	Due: Jan 29, 2018	105

4.9.3 Communication Track

<input type="checkbox"/> ✓ 4.9. Lectures and Lecture Material	Week 1	79
<input type="checkbox"/> ✓ 10. Documenting Scientific Research	Week 1	145
<input type="checkbox"/> ✓ 10.2. Plagiarism	Week 1	146
<input type="checkbox"/> ✓ 1.4. Other Results	Week 1	61
<input type="checkbox"/> ✓ 14.2. Markdown	Week 1	202
<input type="checkbox"/> ✓ 13.1. Basic Emacs	Week 2	195
<input type="checkbox"/> ✓ 10.4. Writing a Scientific Article or Conference Paper	Week 3	149
<input type="checkbox"/> ✓ 11. Introduction to \LaTeX	Week 3	157
<input type="checkbox"/> ✓ 12. Managing Bibliographies	Week 4	177

Evaluation Paper1: Create a paper about a cloud technology with our give class template in the git repository. If a paper is plagiarized you will receive an ‘‘F’’ and it is reported based on University policies.

4.9.4 Theory Track

Introduction to Cloud Computing

<input type="checkbox"/> ✓ 33.1.1. Introduction - Part A	Released, Feb 11, 2018	400
<input type="checkbox"/> ✓ 33.1.2. Introduction - Part B - Defining Clouds I	Released, Feb 11, 2018	400
<input type="checkbox"/> ✓ 33.1.3. Introduction - Part C - Defining Clouds II	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.4. Introduction - Part D - Defining Clouds III	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.5. Introduction - Part E - Virtualization	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.6. Introduction - Part F - Technology Hypecycle I	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.7. Introduction - Part G - Technology Hypecycle II	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.8. Introduction - Part H - IaaS I	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.9. Introduction - Part I - IaaS II	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.10. Introduction - Part J - Cloud Software	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.11. Introduction - Part K - Applications I	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.13. Introduction - Part M - Applications III	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.14. Introduction - Part N - Parallelism	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.15. Introduction - Part O - Storage	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.16. Introduction - Part P - HPC in the Cloud	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.17. Introduction - Part Q - Analytics and Simulation	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.18. Introduction - Part R - Jobs	Released, Feb 11, 2018	404

- ✓ [33.1.19. Introduction - Part S - The Future](#) Released, Feb 11, 2018 404
- ✓ [33.1.20. Introduction - Part T - Security](#) Released, Feb 11, 2018 404
- ✓ [33.1.21. Introduction - Part U - Fault Tolerance](#) Released, Feb 11, 2018 404

Big Data Applications

- ✓ [62. Big Data Use Cases Survey](#) Available Jan 1, 2018, Released Mar 10, 2018 595

REST

Virtual Machines

Containers

Hadoop

Evaluation Paper1: Create a paper about a cloud technology with our give class template in the git repository. If a paper is plagiarized you will receive an “F” and it is reported based on University policies. The paper is in a directory called paper1. All images are in the directory paper1/images, the report is in report.tex, the content is in content.tex. It follows the template we provided. Submission of report.pdf is not allowed. We will create the report and check completeness that way.

4.9.5 Programming Track

Development Environment

- ✓ [18. Virtual Box](#) Week 2 235
- ✓ [15. Linux](#) Week 2 209
- ✗ [16. SSH](#) Week 2 215
- ✗ [17. Github](#) Week 3 225

Evaluation Experiment 1: Create a virtual machine and take a photo with your laptop or computer and the virtual box running on the screen. Showcase the virtual box interface and in non full screen mode at the same time the operating system you run. We v recommend you use Ubuntu.

Python

- ✓ [21. Introduction](#) Week 3 - 4 291
- ✓ [22. Install](#) Week 3 293
- ✓ [23. Language](#) Week 3 303
- ✓ [25. Libraries](#) Week 3 323
- ✗ [31. Cloudmesh Command Shell](#) Week 4 387

Cloud

- ✓ [34.1. Overview of REST](#) Week 2 407
- ✓ [34.3. Rest Services with Eve](#) Week 2 410
- ✓ [34.11. REST Service Generation with Swagger](#) Week 3 426

Assignments

Develop a Big Data related REST service with swagger.

Chose two of them so you can see the difference between the APIs:

- Build a Multi cloud interface to OpenStack
- Build a Multi cloud interface to AWS
- Build a Multi cloud interface to Azure
- Build a Multi cloud interface to Docker/Kubernetes

Evaluation Experiment 1: Create a program in python that identifies a termination criteria for the Secchi disk problem. E.g. at what image can we no longer see the disk? Describe your solution in md and submit to the git repository in a directory called *secchi*. The program is called *secchi.py*, the description is in *README.md*. It uses *cmd5* for creating a command shell that can load the data and analyze it.

Evaluation Assignment 1: Create two a REST service for Big Data with swagger and provide a python implementation from the exported python code to provide real functionality.

Evaluation Assignment 2: Develop cloudmesh extensions to access cloud services from 2 different providers.

Chameleon Cloud and OpenStack

<input type="checkbox"/> ✓ 20. Chameleon Cloud	Week 4	247
<input type="checkbox"/> ✓ 20.5. Charge Rates	Week 4	253
<input type="checkbox"/> ✓ 20.3. Hardware	Week 4	249
<input type="checkbox"/> ✓ 20.4. Getting Started	Week 4	251
<input type="checkbox"/> ✓ 20.7. Horizon Graphical User Interface	Week 5	261
<input type="checkbox"/> ✓ 20.6. OpenStack Virtual Machines	Week 4	254
<input type="checkbox"/> ✗ 20.8. HEAT	Optional	268
<input type="checkbox"/> ✗ 20.9. Bare Metal	Optional	284



5. E616/I524: Advanced Cloud Computing

This class is also known under the name *I524 Big Data Applications and Open Source Software*.

We present the syllabus for the E616 course taught at Indiana University that uses in part the material presented in this document. The information includes the section or chapter number, the name of the chapter or section, and the timeframe when it is recommended to work through the material and the page number where to find the material in this document. Please note that we will update the document throughout the semester thus, page numbers will change.

5.1 Course Description

This course describes Cloud 3.0 in which DevOps, Microservices, and Function as a Service is added to basic cloud computing. The discussion is centered around the Apache Big Data Stack and a major student project aimed at demonstrating integration of cloud capabilities.

5.2 Course pre-requisites

Python will be used as a Programming Language. It is expected that you know a programming language. ENGR-E516 or an introduction to cloud computing is recommended. Studnets are expected to have access to a computer on which they can execute Linux easily. As the OS requirements have recently increased we recommend a computer with 8GB main memory and the ability to run virtualbox and/or containers. If it turns out your machine is not capable enough we attempt to provide access to IU linux machines.

5.3 Course Registration

I524 and E616 are identical. In this courses we will be focussing on Advanced Cloud Computing and Big Data Applications and Analytics.

Intelligent Systems Engineering Residential:

```

ENGR-E 616  ADVANCED CLOUD COMPUTING (3 CR)
***** RSTR      ARR          ARR      ARR      Von Laszewski G
  Above class taught online
  Above class open to graduates only
  Discussion (DIS)
33697 RSTR      09:30A-10:45A  F        I2 150  Von Laszewski G

```

Info Residential:

```

INFO-I 524  BIG DATA SOFTWARE AND PROJECTS (3 CR)
***** RSTR      ARR          ARR      ARR      Von Laszewski G
  Above class open to graduates only
  Above class taught online
  Discussion (DIS)
13053 RSTR      09:30A-10:45A  F        I2 150  Von Laszewski G

```

Info Online:

```

INFO-I 524  BIG DATA SOFTWARE AND PROJECTS (3 CR)
13054 RSTR      ARR          ARR      ARR      Von Laszewski G
  Above class open to graduates only
  This is a 100% online class taught by IU Bloomington. No
  on-campus class meetings are required. A distance education
  fee may apply; check your campus bursar website for more
  information

```

5.4 Teaching and learning methods

- Lectures
- Assignments including specific lab activities
- Final project
- Class will use software mainly written in Python and Linux Shell.

5.5 Covered Topics

The class will cover a number of topics as part of tracks that are executed in parallel throughout the class. Although we assume that at a graduate course level the communication track has already been covered elsewhere, we made sure we also include it here in case you did not yet have such experiences.

Please note that we may change the order of the lectures. This is our current plan

- **Week 1-2: Introduction** Overview and status of cloud computing
- **Week 2: Cloud REST Services** Introduction to REST services to develop cloud services.
- **Week 3: Big Data Reference Architecture:** Using REST services to implement Big Data Services.
- **Week 4: Cloud and Big Data Applications:** Introduction to the Apache Big Data Stack. Selective presentation of the members of the Big Data Application set
- **Week 5: New Cloud Big Data Application Technologies:** Students will explore the Apache Web page and report on them. Continuation of earlier assigned homework, including paper.
- **Week 6: DevOps:** Introduction to DevOps with Dockerfile and ansible
- **Week 7-14: Building a Kubernetes Cluster:** Residential students will be building a kubernetes cluster with 5 servers. Online students could do that also, but need to simulate the cluster instead of using real hardware as it requires physical access to the hardware.

- **Week 7: Virtual Machines:** Introduction to OpenStack
- **Week 8-9: Containers:** Introduction to Container technology
- **Week 10: Advanced Containers:** Building clusters with containers.
- **Week 11: Cloud 3.0:** Microservices, Events, Functions
- **Week 12-14: Project:** Delivery of a reproducible substantial student project (you are allowed to substantially enhance a project that you started from other cloud classes you took with us. Please ask.)

5.6 Student learning outcomes

When students complete this course, they should be able to:

- Have an advanced understanding of issues involved in designing and applying modern cloud technologies using the latest developments.
- Gain hands-on laboratory experience.
- Understand the Apache Big Data Software Stack.
- Apply knowledge of mathematics, science, and engineering.
- Understand research challenges and important application areas of clouds
- Have advanced skills in teamwork with peers.
- Be able to use DevOps technologies.

5.7 Grading

Grade Item	Percentage
Assignments	40%
Final Project	50%
Participation	10%

5.8 Representative bibliography

1. Cloud Computing for Science and Engineering By Ian Foster and Dennis B. Gannon
 - <https://mitpress.mit.edu/books/cloud-computing-science-and-engineering>
2. Machine to machine protocols <https://en.wikipedia.org/wiki/MQTT>
3. Cloud software systems <http://hpc-abds.org/kaleidoscope/>
4. Software Defined Networks https://en.wikipedia.org/wiki/Software-defined_networking
5. There are a huge number of other web resources
6. (This document) **Handbook of Clouds and Big Data**, Gregor von Laszewski, Geoffrey C. Fox, and Judy Qiu, Fall 2017, <https://tinyurl.com/vonLaszewski-handbook>
7. **Use Cases in Big Data Software and Analytics Vol. 1**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v1.pdf>
8. **Use Cases in Big Data Software and Analytics Vol. 2**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v2.pdf>
9. **Use Cases in Big Data Software and Analytics Vol. 3**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/vonLaszewski-projects-v3>
10. (Draft) **Big Data Software Vol 1.**, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper1/proceedings.pdf>
11. (Draft) **Big Data Software Vol 2.**, Gregor von Laszewski, Spring 2017,
 - <https://github.com/cloudmesh/sp17-i524/blob/master/paper2/proceedings.pdf>

12. (Draft) **Big Data Projects**, Gregor von Laszewski, Spring 2017,
 • <https://github.com/cloudmesh/sp17-i524/blob/master/project/projects.pdf>


5.9 Lectures and Lecture Material

Warning

This section will change and be updated throughout the semester

Indiana University

To start the class at IU, we require you to watch the following video:

Class Management - Overview (39:54) 

Teaching Cloud and Big Data - Overview (39:54) 

5.9.1 Class Graph E616 and I524

We present the Class outline also in a graph representation in Figure 5.1. Hyperlinks to the specific sections are included

TODO: TA: add hyperlinks

5.9.2 Assignments

All assignments are summarized in the Section 7.2

The current released assignments include:

- | | | | |
|---|---------------------------|-------------------|-----|
| <input type="checkbox"/> ✓ 7.2.1. Bio Post | Exercise 7.13, 7.14, 7.15 | Due: Jan 22, 2018 | 104 |
| <input type="checkbox"/> ✓ 7.2.3. Big Data Collaboration | | Due: Jan 29, 2018 | 104 |
| <input type="checkbox"/> ✓ 7.2.4. New Technology List | | Due: Jan 29, 2018 | 105 |
| <input type="checkbox"/> ✓ 7.2.5. New Technology Abstract | | Due: Jan 29, 2018 | 105 |

5.9.3 Communication Track

- | | | |
|---|--------|-----|
| <input type="checkbox"/> ✓ 5.9. Lectures and Lecture Material | Week 1 | 88 |
| <input type="checkbox"/> ✓ 10. Documenting Scientific Research | Week 1 | 145 |
| <input type="checkbox"/> ✓ 10.2. Plagiarism | Week 1 | 146 |
| <input type="checkbox"/> ✓ 1.4. Other Results | Week 1 | 61 |
| <input type="checkbox"/> ✓ 14.2. Markdown | Week 1 | 202 |
| <input type="checkbox"/> ✓ 13.1. Basic Emacs | Week 2 | 195 |
| <input type="checkbox"/> ✓ 10.4. Writing a Scientific Article or Conference Paper | Week 3 | 149 |
| <input type="checkbox"/> ✓ 11. Introduction to L ^A T _E X | Week 3 | 157 |
| <input type="checkbox"/> ✓ 12. Managing Bibliographies | Week 4 | 177 |

Evaluation Paper1: Create a paper about a cloud technology with our give class template in the git repository. If a paper is plagiarised you will receive an “F” and it is reported based on

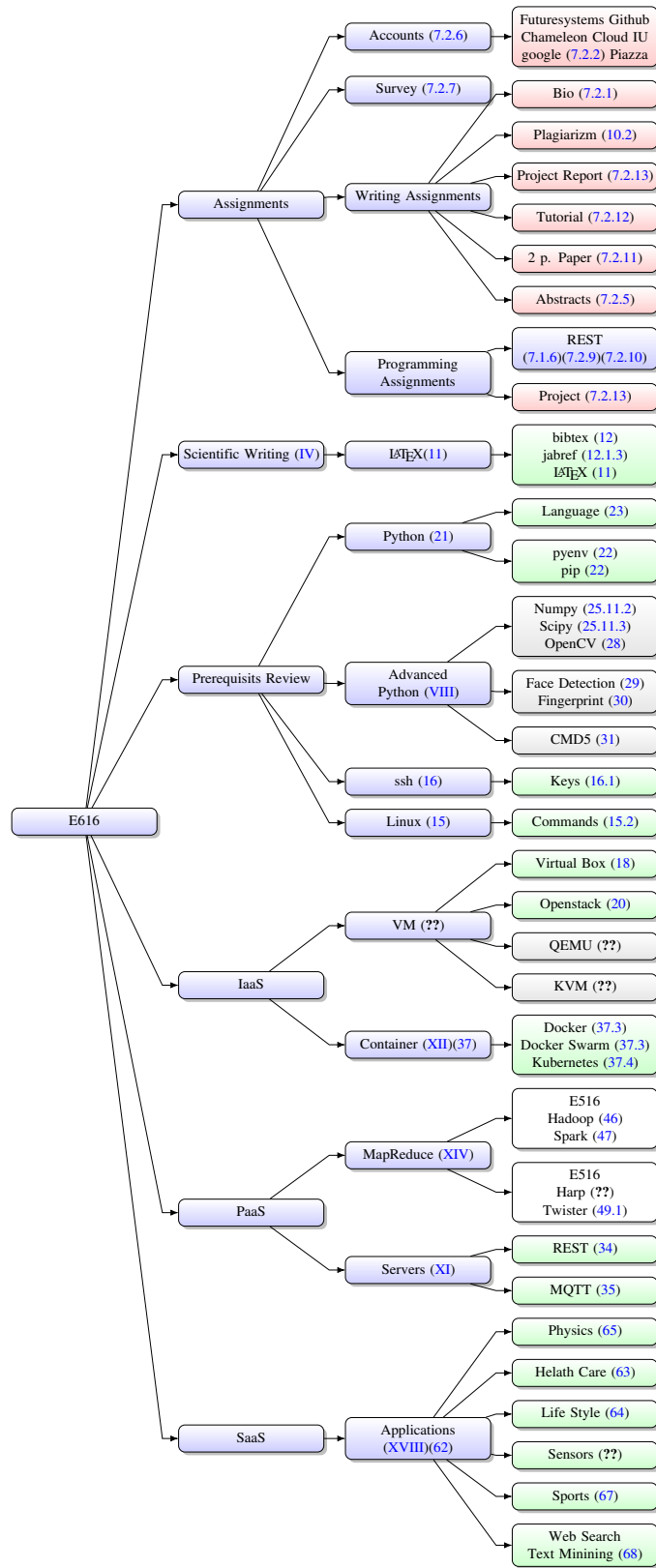


Figure 5.1: E616 an I524 class structure.

University policies.

5.9.4 Theory Track

Introduction to Cloud Computing

<input type="checkbox"/> ✓ 33.1.1. Introduction - Part A	Released, Feb 11, 2018	400
<input type="checkbox"/> ✓ 33.1.2. Introduction - Part B - Defining Clouds I	Released, Feb 11, 2018	400
<input type="checkbox"/> ✓ 33.1.3. Introduction - Part C - Defining Clouds II	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.4. Introduction - Part D - Defining Clouds III	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.5. Introduction - Part E - Virtualization	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.6. Introduction - Part F - Technology Hypecycle I	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.7. Introduction - Part G - Technology Hypecycle II	Released, Feb 11, 2018	401
<input type="checkbox"/> ✓ 33.1.8. Introduction - Part H - IaaS I	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.9. Introduction - Part I - IaaS II	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.10. Introduction - Part J - Cloud Software	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.11. Introduction - Part K - Applications I	Released, Feb 11, 2018	402
<input type="checkbox"/> ✓ 33.1.13. Introduction - Part M - Applications III	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.14. Introduction - Part N - Parallelism	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.15. Introduction - Part O - Storage	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.16. Introduction - Part P - HPC in the Cloud	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.17. Introduction - Part Q - Analytics and Simulation	Released, Feb 11, 2018	403
<input type="checkbox"/> ✓ 33.1.18. Introduction - Part R - Jobs	Released, Feb 11, 2018	404
<input type="checkbox"/> ✓ 33.1.19. Introduction - Part S - The Future	Released, Feb 11, 2018	404
<input type="checkbox"/> ✓ 33.1.20. Introduction - Part T - Security	Released, Feb 11, 2018	404
<input type="checkbox"/> ✓ 33.1.21. Introduction - Part U - Fault Tolerance	Released, Feb 11, 2018	404

Big Data Applications

<input type="checkbox"/> ✓ XVIII. REST Services	Available Jan 1. 2018, Released Mar 10, 2018	595
---	--	-----

This section includes a number of lectures in various application domains

<input type="checkbox"/> ✓ 62. Big Data Use Cases Survey	Available Jan 1. 2018, Released Mar 10, 2018	595
<input type="checkbox"/> ✓ 63. Health Informatics	Available Jan 1. 2018, Released Mar 10, 2018	605
<input type="checkbox"/> ✓ 64. e-Commerce and LifeStyle	Available Jan 1. 2018, Released Mar 10, 2018	611
<input type="checkbox"/> ✓ 65. Physics	Available Jan 1. 2018, Released Mar 10, 2018	617
<input type="checkbox"/> ✓ ??.	Available Jan 1. 2018, Released Mar 10, 2018	??
<input type="checkbox"/> ✓ 67. Sports	Available Jan 1. 2018, Released Mar 10, 2018	629
<input type="checkbox"/> ✓ ??.	Available Jan 1. 2018, Released Mar 10, 2018	??

REST

Virtual Machines

Containers

Hadoop

Evaluation Paper1: Create a paper about a cloud technology with our give class template in the git repository. If a paper is plagiarized you will receive an “F” and it is reported based on

University policies. The paper is in a directory called `paper1`. All images are in the directory `paper1/images`, the report is in `report.tex`, the content is in `content.tex`. It follows the template we provided. Submission of `report.pdf` is not allowed. We will create the report and check completeness that way.

5.9.5 Programming Track

Development Environment

<input type="checkbox"/> ✓ 18. Virtual Box	Week 2	235
<input type="checkbox"/> ✓ 15. Linux	Week 2	209
<input type="checkbox"/> ✗ 16. SSH	Week 3	215
<input type="checkbox"/> ✗ 17. Github	Week 3	225

Evaluation Experiment 1: Create a virtual machine and take a photo with your laptop or computer and the virtual box running on the screen. Showcase the virtual box interface and in non full screen mode at the same time the operating system you run. We recommend you use Ubuntu.

Python

<input type="checkbox"/> ✓ 21. Introduction	Week 3 - 4	291
<input type="checkbox"/> ✓ 22. Install	Week 3	293
<input type="checkbox"/> ✓ 23. Language	Week 3	303
<input type="checkbox"/> ✓ 25. Libraries	Week 3	323
<input type="checkbox"/> ✗ 31. Cloudmesh Command Shell	Week 4	387
<input type="checkbox"/> ✗ 26. Numpy		339
<input type="checkbox"/> ✗ 27. Scipy		343
<input type="checkbox"/> ✗ 28. OpenCV		349
<input type="checkbox"/> ✗ 28.5. Secchi Disk		353

Evaluation Experiment 1: Create a program in python that identifies a termination criteria for the Secchi disk problem. E.g. at what image can we no longer see the disk? Describe your solution in md and submit to the git repository in a directory called `secchi`. The program is called `secchi.py`, the description is in `README.md`. It uses `cmd5` for creating a command shell that can load the data and analyze it.

5.9.6 DevOps

- ssh for DevOps
- Ansible
- Dockerfile

5.9.7 Cloud

<input type="checkbox"/> ✓ 34.1. Overview of REST	Week 2	407
<input type="checkbox"/> ✓ 34.3. Rest Services with Eve	Week 2	410
<input type="checkbox"/> ✓ 34.11. REST Service Generation with Swagger	Week 3	426

- OpenAPI REST Service
- OpenAPI Big Data Services
- Deploy cloud services with Ansible

Chameleon Cloud and OpenStack

<input type="checkbox"/> ✓ 20. Chameleon Cloud	Week 4	247
<input type="checkbox"/> ✓ 20.5. Charge Rates	Week 4	253
<input type="checkbox"/> ✓ 20.3. Hardware	Week 4	249
<input type="checkbox"/> ✓ 20.4. Getting Started	Week 4	251
<input type="checkbox"/> ✓ 20.7. Horizon Graphical User Interface	Week 5	261
<input type="checkbox"/> ✓ 20.6. OpenStack Virtual Machines	Week 4	254
<input type="checkbox"/> ✗ 20.8. HEAT	Optional	268
<input type="checkbox"/> ✗ 20.9. Bare Metal	Optional	284

5.10 Containers

- Introduction to Docker - Docker File
- Advanced Docker - Docker Swarm - Kubernetes
- Deploy cloud services with Docker/Kubernetes
- Build a Raspberry Pi based Kubernetes cluster with 5 nodes (residential student can work on this in teams up to 5 students, we have 50 Raspberry Pi's)
- Benchmark a 144 node Raspberry PI kubernetes cluster after deploying it (open class work)



6. E222: Intelligent Systems Engineering II

6.1 Course Description

In this undergraduate course students will be familiarized with different specific applications and implementations of intelligent systems and their use in desktop and cloud solutions.

6.2 Course pre-requisites

One programming language, Intelligent Systems Engineering I or equivalent

6.3 Course Registration

```
ENGR-E 222 INTELLIGENT SYSTEMS II (3 CR)
      ***** RSTR      02:30P-03:45P   TR      Luddy 4063 Fox
      Laboratory (LAB)
E 222 : P - ENGR-E 221
      31434 RSTR      05:45P-06:35P   R      Luddy 4063 Fox
      Above class for Intelligent Systems Engineering students
```

6.4 Teaching and learning methods

- Lectures
- Assignments including specific lab activities
- Final project

6.5 Covered Topics

The topics covered in this class include.

- Introduction to REST: Theory and Practice - develop a REST service
- Introduction to Clouds: Theory and Practice - create via a program virtual machines and start on them the REST service
- Introduction to Kubernetes: Theory and Practice - create a container that runs a REST service
- Introduction to Advanced AI: Integrate your AI engine into a REST service and run on a cloud and in Kubernetes
- Introduction to Hadoop: Theory and Practice - Run Hadoop in a container; run hadoop on a futuresystems cluster
- Edge Computing: Theory and Practice - Integrate Sensordata into Cloud Services via REST and MQTT

6.6 Student learning outcomes

When students complete this course, they should be able to:

- Have an elementary understanding of issues involved in Cloud Computing as part of the intelligent systems effort.
- Gain hands-on laboratory experience with several examples.
- Apply knowledge of mathematics, science, and engineering
- Understand research challenges and important issues with Software Defined Systems.
- Have advanced skills in teamwork with peers.
- Have theoretical and practical knowledge about REST, Clouds, Containers, and Edge Computing.

6.7 Grading

Grade Item	Percentage
Assignments	60%
Final Project	30%
Participation	10%

6.8 Representative bibliography

1. Cloud Computing for Science and Engineering By Ian Foster and Dennis B. Gannon
 - <https://mitpress.mit.edu/books/cloud-computing-science-and-engineering>
2. (This document) **Handbook of Clouds and Big Data**, Gregor von Laszewski, Geoffrey C. Fox, and Judy Qiu, Fall 2017, <https://tinyurl.com/vonLaszewski-handbook>
3. **Use Cases in Big Data Software and Analytics Vol. 1**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v1.pdf>
4. **Use Cases in Big Data Software and Analytics Vol. 2**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/cloudmesh/vonLaszewski-i523-v2.pdf>
5. **Use Cases in Big Data Software and Analytics Vol. 3**, Gregor von Laszewski, Fall 2017, <https://tinyurl.com/vonLaszewski-projects-v3>
6. (Draft) **Big Data Software Vol 1**, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper1/proceedings.pdf>
7. (Draft) **Big Data Software Vol 2**, Gregor von Laszewski, Spring 2017,
 - <https://github.com/cloudmesh/sp17-i524/blob/master/paper2/proceedings.pdf>
8. (Draft) **Big Data Projects**, Gregor von Laszewski, Spring 2017,
 - <https://github.com/cloudmesh/sp17-i524/blob/master/project/projects.pdf>

6.9 Lectures and Lecture Material

6.9.1 Class Graph E516

We present the Class outline als in a graph representation in Figure 6.1. Hyperlinks to the specific sections are included

TODO: TA: add hyperlinks

6.9.2 Proposed Weekly Agenda

Warning

The following information is related to our initial plans of presenting material for E222 by Week. Please note that this could change. These are proposed topics and we will update the Handbook accordingly from week to week.

6.9.3 Week 1. Administration

We have explained how to use piazza which we will be using for class communication. Students that missed that lecture are responsible for working with TAs to catch up.

6.9.4 Weeks 1, 2 and 4. Introdcution to Cloud Computing

A number of presentations have been posted to introduce you to ths Class. The following lectures are the video lectures from weeks 1-2.

<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.1. Introduction - Part A	Released, Feb 1, 2018	400
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.2. Introduction - Part B - Defining Clouds I	Released, Feb 1, 2018	400
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.3. Introduction - Part C - Defining Clouds II	Released, Feb 1, 2018	401
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.4. Introduction - Part D - Defining Clouds III	Released, Feb 1, 2018	401
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.5. Introduction - Part E - Virtualization	Released, Feb 1, 2018	401
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.6. Introduction - Part F - Technology Hypecycle I	Released, Feb 1, 2018	401
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.7. Introduction - Part G - Technology Hypecycle II	Released, Feb 1, 2018	401
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.8. Introduction - Part H - IaaS I	Released, Feb 1, 2018	402
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.9. Introduction - Part I - IaaS II	Released, Feb 1, 2018	402
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.10. Introduction - Part J - Cloud Software	Released, Feb 1, 2018	402
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.11. Introduction - Part K - Applications I	Released, Feb 1, 2018	402
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.13. Introduction - Part M - Applications III	Released, Feb 1, 2018	403
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.14. Introduction - Part N - Parallelism	Released, Feb 1, 2018	403
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.15. Introduction - Part O - Storage	Released, Feb 1, 2018	403
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.16. Introduction - Part P - HPC in the Cloud	Released, Feb 1, 2018	403
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.17. Introduction - Part Q - Analytics and Simulation	Released, Feb 1, 2018	403
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.18. Introduction - Part R - Jobs	Released, Feb 1, 2018	404
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.19. Introduction - Part S - The Future	Released, Feb 1, 2018	404
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.20. Introduction - Part T - Security	Released, Feb 1, 2018	404
<input type="checkbox"/>	<input checked="" type="checkbox"/>	33.1.21. Introduction - Part U - Fault Tolerance	Released, Feb 1, 2018	404

This introduction has been continued in Week 4 of the class by Geoffrey posting additional lectures.

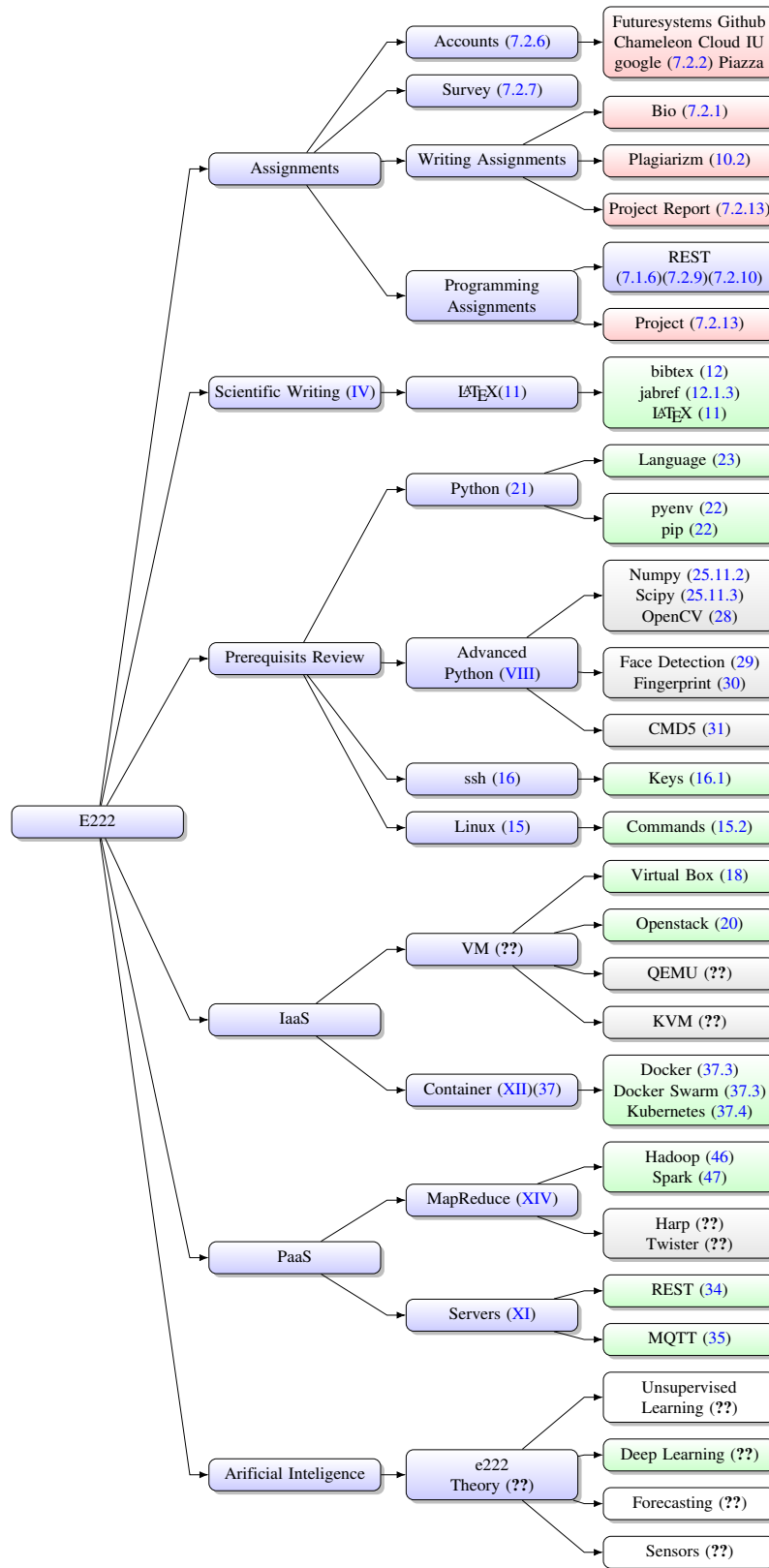


Figure 6.1: E222 class structure **THIS GRAPH IS WRONG AND WILL BE MODIFIED.**

TODO: Tyler: include the links to Geoffreys lectures he presented, didn't see the slides presented on Thursday Feb 1,2018 about parallel computing, Amdahls Law I beleive was the first slide

6.9.5 Week 3. REST for Cloud computing

We will be starting the class with introducing you to REST services that provide a foundation for setting up services in the cloud and to intercat with these services. As part of this class we will be revisiting the REST services and use them to deploy them on a cloud as well as develop our own AI based rest services in the second half of the class.

To get you started you need to read the follwoing sections:

An introduction to rest is provided in Section 34. It also provides a video recording of the material that was presented in class.

✓ [34.1. Overview of REST](#) 407

Next, we present you with information on how to easily create a rest service with FlaskRESTFUL a libraruy that makes the creation of web services in python using an object oriented approach easy. A Lab was held that introduced you to developing such a service

✓ [34.2. Flask RESTful Services](#) 408

A number of contributions from students have since been made including the development effort for this lab. Links to this work can be found at

- <https://github.com/cloudmesh-community/hid-sp18-505/tree/master/rest>
- <https://github.com/cloudmesh-community/hid-sp18-409/tree/master/rest>

Next, we introduced you to Python Eve which ia a framework that allows you to define rest services with schemas. This is important as it first allows you to easily define them while having just to do a very minimal set of programming. Second, it allows you via Eve to make sure you define a well designed data model that you can communicate to the users of the REST service.

✓ [34.3. Rest Services with Eve](#) 410

In the Lab that may spawn multiple weeks you will be developing a Flask or Eve rest service. The REST service retrieves information form your computer and exposes it to the client. You can chose what you like to present, but we want that all students in class do a different information. TAs are working with you which information you expose. A detailed Assignment is posted and coordinated in piazza about this. YOu will be writing two services. One that uses flask features, the other one that uses a database schema using Eve. The entire class can openly collaborate with each other on this task. The code is to be checked in into your github repository. Information of interrest include memory available of the computer, cpu type and so on. This is not read from a text file but life queried.

TODO: Tyler:Describe how each student gets a unique assignment. Coordinate that assignment.

The link to the assignmnet in piazza is below, directions for each student will come out the week of February 4, 2018.

- <https://piazza.com/class/jc9dcfnbi045kv?cid=27>

6.9.6 Weeks 2-3. Setting up your development environment

It is important that you have a development environment to conduct the class assignments. We recommend that you use virtual box and use ubuntu. We have provided an extensive set of material for you to achieve this. Please consult additional resources form the Web

The material includes:

TODO: Tyler: please include links to the sections

- Install virtual box
- Install ubuntu on the virtual box
- ✓ [18. Virtual Box](#) Week 2 235
- Install an editor to develop python programs. We recommend pyCharm or simply emacs.
- ✓ [13.1. Basic Emacs](#) Weeks 2-3 195
- Using pyenv for multi version python installs
- ✓ [22.1.1. Managing Multiple Python Versions with Pyenv](#) Weeks 2-3 293

Technologies covered: piazza, git, pycharm, virtualbox, pyenv, python

Exersise: Continue to work on REST service.

6.9.7 Week 5. Introduction to simple Containers

We will be providing an introduction to containers and container technologies. Excercises will include to run the REST services that we developed earlier to start in containers and utilize them.

- ✓ [37.1. Motivation - Microservices](#) Released Feb 11, 2018 451
- ✓ [37.2. Motivation - Serverless Computing](#) Released Feb 11, 2018 451

6.9.8 Week 6. Introduction to Container Clusters

We will be providing you with an introduction on how to not use one server, but multiple servers to run containers on. This includes docker swarm, kubernetes, (maybe mesos if time allows). Exercises include the deployment of minikube that enables you to run kubernetes on your computers. Alternatively access to a docker swarm cluster may be provided.

- ✓ [37.3. Docker](#) Released Feb 11, 2018 451
- ✓ [37.4. Docker and Kubernetes](#) Released Feb 11, 2018 452

6.9.9 Week 7. Map Reduce

In this section we will introduce you to the concept of Map reduce. We will discuss systems such as Hadoop and Spark and how they differ. You will be deploying via a container hadoop on your machine and use it to gain hands on experience.

6.9.10 Week 8. Overview of Cloud Services

We will be introducing you how to use Cloud services offered via a number of Cloud providers. Topics covered include: overview of AWS, overview of Openstack, libcloud, and boto

6.9.11 Week 9. Multi cloud environments

We will teach you how to create a multi cloud shell while leveraging an abstract programming interface to easily switch between multiple clouds. You can practically participate in helping to develop interfaces to AWS, Azure, and OpenStack. As you have also worked with containers, you can develop such interfaces also for containers including frameworks such as kubernetes. We will be using libcloud to simplify the abstraction.

6.9.12 Week 10. Cloud Data and Applications

We will cover a number of Application examples for Cloud computing. In the second part we will focus on CCloud Data Services and how we access data on the cloud. Exercises will include moving data between data services. This includes your own computer, box and google which both are offered at IU.

6.9.13 Other weeks

All excercises in these weeks will develop REST services that expose machine leraning algorithms as service. Data will be either passed along directly through parameters to the call, or on case of large data a URL to the data source. The lessons from the previous weeks will be helping you to achive this. It is not sufficient to just run the algorithms, but you must be integrating them into a REST service.

Other weeks are not yet included here but will cover Artificial Intelligence.

6.9.14 Assignments

All assignments are summarized in the Section 7.1 but are posted earlier in piazza.com in the pinned section.

6.9.15 Communication Track

<input type="checkbox"/> ✓ 10. Documenting Scientific Research	Week 1	145
<input type="checkbox"/> ✓ 10.2. Plagiarism	Week 1	146
<input type="checkbox"/> ✓ 1.4. Other Results	Week 1	61
<input type="checkbox"/> ✓ 14.2. Markdown	Week 1	202
<input type="checkbox"/> ✓ 13.1. Basic Emacs	Week 2	195
<input type="checkbox"/> ✓ 10.4. Writing a Scientific Article or Conference Paper	Week 3	149
<input type="checkbox"/> ✓ 11. Introduction to L ^A T _E X	Week 3	157
<input type="checkbox"/> ✓ 12. Managing Bibliographies	Week 4	177

Evaluation Paper1: Create a paper about a cloud technology with our give class template in the

git repository. If a paper is plagiarised you will receive an ‘‘F’’ and it is reported based on University policies.

6.9.16 Theory Track

- IaaS - OpenStack
- PaaS - Hadoop
- SaaS - SaaS with REST

Evaluation Paper1: Create a paper about a cloud technology with our give class template in the git repository. If a paper is plagiarized you will receive an ‘‘F’’ and it is reported based on University policies. The paper is in a directory called paper1. All images are in the directory paper1/images, the report is in report.tex, the content is in content.tex. It follows the template we provided. Submission of report.pdf is not allowed. We will create the report and check completeness that way.

6.9.17 Programming Track

Development Environment

<input type="checkbox"/> ✓ 18. Virtual Box	Week 2	235
<input type="checkbox"/> ✓ 15. Linux	Week 2	209
<input type="checkbox"/> ✗ 16. SSH	Week 3	215
<input type="checkbox"/> ✗ 17. Github	Week 3	225

Python

<input type="checkbox"/> ✓ 21. Introduction	Week 3 - 4	291
<input type="checkbox"/> ✓ 22. Install	Week 2	293
<input type="checkbox"/> ✓ 23. Language	Week 2	303
<input type="checkbox"/> ✓ 25. Libraries	Week 3	323
<input type="checkbox"/> ✗ 31. Cloudmesh Command Shell	Week 4	387

Cloud

<input type="checkbox"/> ✓ 34.1. Overview of REST	Week 2	407
<input type="checkbox"/> ✓ 34.3. Rest Services with Eve	Week 2	410
<input type="checkbox"/> ✓ 34.11. REST Service Generation with Swagger	Week 3	426

- Build a Rest Service
- Build a program to create VMs on an OpenStack cloud.



7. Assignments

The assignments are listed in chronological order. All assignments posted here are supposed to be conducted by you. There is a slight delay between assignments posted in piazza and the assignments in this section as TAs need to be integrating them. Typical delay is one business day. Business days are defined as Mon-Fri 9am-5pm EST. Thus, it is often better for those working on the weekends to visit piazza.com instead. The assignments are ordered by class. Please focus on conducting the assignments listed either here on in piazza. Just as any textbook has many exercises, we are providing selected exercises for you as assignments. Not every exercise mentioned in a chapter has to be done.

Indiana University

There can not be any confusion which assignments have been issued, as they are all pinned in piazza and you can visit piazza.com to find out which are assigned and listed there.

If you use a calendar system, it is in your responsibility to manage it in such a system. This could include Google, Outlook, CANVAS, and may others. Naturally you can also use to do lists that you can manage as part of your github repository issues, once you have access to. This is probably the preferred method as it allows you to add the tasks yourself and you can check them on and off as you have conducted the assignments.

This way we teach you how to use open source technologies to coordinate your own work with your own time management tools and constraints in mind. This is in contrast to just using CANVAS as CANVAS does not support open source developments in teams. Furthermore it is unlikely that you will ever use CANVAS after you graduate. We rather like to have you use systems that you use after you graduate. However, if you still like to use CANVAS for alerting you, that is entirely up to you as you can add assignments yourself to CANVAS.

Indiana University

All assignments are due Monday morning at 9:00 am est. No exceptions unless otherwise specified.

7.1 Assignments E222

All assignments posted here are supposed to be conducted by you.

7.1.1 Bio Post

Exercise 7.1 Bio Post on Piazza. Please post a formal bio to piazza ■

Exercise 7.2 Bio Post in Google doc. due Feb 5, 2018.

After you have posted it to piazza copy your updated formal bios into the following document. Make sure you use 3rd person and stay formal. This is a formal bio. Comment on the effectiveness of using the cloud service for this task. At the end of the document. This assignment does not replace the post of the bio to piazza, it is used to gather all bios in one document and to evaluate if google docs is a good tool for this kind of task. Remember we have lots of students and google is used often just with small groups.

[E222 Link to google doc ↗](#) ■

7.1.2 Cloud Accounts

Exercise 7.3 IU Google Services. due Feb 5, 2018

This assignment is only for those that do not yet have access to our google documents This assignment does not have to be conducted for anyone that has access to our google documents for bios, and the technology list

- What is the difference between umail.iu.edu and iu.edu? Tip: the answer is provided in the IU knowledge base
- Login via the iu.edu account and not the umail.iu.edu account to google and open the document for the bio. Paste the bio into the document.
- Explain why IU has two different google services and logins. As we use cloud in this class, it is important to understand this and what implication this has. This is not just an assignment to give you access to the service, but to make you think why this works like this.
- Can you imagine a different way this ought to work? ■

7.1.3 Account Creation

Exercise 7.4 Account Creation: github.com

If you do not have a github.com account, go to github.com and apply for a <https://github.com> account. Write down your account name and remember the password. You will need the account for upcoming assignments.

Exercise 7.5 Account Creation: futuresystems.org

If you do not have a futuresystems.org account go to <https://portal.futuresystems.org/user/register> and apply for an account. Write down your account name and remember the password. You will need the account for upcoming assignments.

Exercise 7.6 Account Creation: chameleoncloud.org.

If you do not have a chameleon cloud account please go to <https://www.chameleoncloud.org> and apply for an account only. Do not apply for a project. Write down your account name and remember the password. You will need the account for upcoming assignments.

Exercise 7.7 Account Collection Form due Feb 5, 2018

Fill out the form so we can activate your accounts. You will need the account for upcoming assignments.

[E222 Account Collection Form](#) ↗

7.1.4 Entry Survey**Exercise 7.8 Entry Survey**

Please fill out the following survey ASAP as it will determine some of the class material we prepare based on your feedback. The survey is really simple and can be finished in under 5 minutes. <https://goo.gl/forms/Q04FW9eBM7eyL0Lv1>

7.1.5 Account Setup**Exercise 7.9 Github Setup** due Feb, 2018

This exercise will assist you with your github account setup. Please find the README.yml file in the link below and copy it into your github repository and fill out with your information. <https://github.com/cloudmesh-community/hid-sample/blob/master/README.yml> Remove the README.md file. After your .yml file is correctly setup you now need to put your bio in your repository, naming it like so bio-lastname-firstname.tex where lastname is your lastname and so on. Put the .tex file in your repository and use the latex function subsection{lastname,firstname} followed by the text of your bio.

7.1.6 Eve REST Service

Exercise 7.10 Read the Sections Overview of REST (Section 34 and Eve 34.3).

Exercise 7.11 Assignment catch up: please look at all previous assignments and do them.

Exercise 7.12 AI cloud service: This assignment will evolve, therefore we just provide the link to it in piazza. Please visit it:

<https://piazza.com/class/jc9dcfnbi045kv?cid=34> ■

7.2 Assignments E516, I524, E616

7.2.1 Bio Post

Exercise 7.13 Bio Post on Piazza. Please post a formal bio to piazza ■

Exercise 7.14 Bio Post in Google doc. due Feb 5, 2018

After you have posted it to piazza copy your updated formal bios into the following document. Make sure you use 3rd person and stay formal. This is a formal bio. Comment on the effectiveness of using the cloud service for this task. At the end of the document. This assignment does not replace the post of the bio to piazza, it is used to gather all bios in one document and to evaluate if google docs is a good tool for this kind of task. Remember we have lots of students and google is used often just with small groups.

[E516 Link to google doc ↗](#) ■

7.2.2 IU Google Services

Exercise 7.15 IU Google Services: due Feb 5, 2018

This assignment is only for those that do not yet have access to our Google documents This assignment does not have to be conducted for anyone that has access to our Google documents for bios, and the technology list

- What is the difference between uemail.iu.edu and iu.edu? Tip: the answer is provided in the IU knowledge base
- Login via the iu.edu account and not the uemail.iu.edu account to google and open the document for the bio. Paste the bio into the document.
- Explain why IU has two different google services and logins. As we use cloud in this class, it is important to understand this and what implication this has. This is not just an assignment to give you access to the service, but to make you think why this works like this.
- Can you imagine a different way this ought to work?

7.2.3 Big Data Collaboration

Exercise 7.16 Big data and collaboration. due Feb 2018

The purpose of this assignment is multifold; test the ability of Google docs to be used in collaborative fashion by more than a small group and report on the experience. Good Things and bad things, learn on how to use Google docs with headings and table of contents learn how to gather resources quickly with hyperlinks to web resources or articles and translate them into formal academic references. Most importantly convey some very important feature of big

data. Contribute this into the handbook for everyone's benefit (done by TAs).

Task: Your task is to identify Big Data size related articles and Web resources and produce a historical development of the growth of this data

[E516 Link to google doc ↗](#)

7.2.4 New Technology List

Exercise 7.17 Technology List due Jan 29, 2018

The handbook contains a large number of technologies to which an abstract is provided. Your task is to identify FIRST not to do an abstract but to collaboratively gather a LIST of new technologies that are important in Cloud and Big Data. We suggest doing this in a google docs document first. Write Lastname, Firstname, class id behind the technology so we know who contributed it. Indicate also if commercial, or open source, We are mostly interested in open source activities. Keep the list sorted by alphabet. Use a bullet so formatting is preserved

[New Technology List ↗](#)

Example: OpenWhisk, <https://openwhisk.apache.org/>, open source, Gregor von Laszewski, e616

7.2.5 New Technology Abstract

Exercise 7.18 Technology Abstract due Feb 5, 2018

We have gathered with the technology list <https://piazza.com/class/jbkvbp3ed3m2ez?cid=50> a number of technologies that are not yet covered in the handbook or need improvement in the handbook.

The TAs will be selecting about 5 technologies for each student. Each student will write high-quality non-plagiarized abstracts which bibtex references.

Learning outcomes:

- Identify how to not plagiarize
- Work in a large team (with coordination by TAs)
- Use bibtex and jabref for reference management which you will be using for your final paper
- Find new trends in big data and cloud computing

Exercise 7.19 Technology Abstract upload to GitHub due Feb 26, 2018

Please review the plagiarism and quoting guidelines chapter in the handbook.

✓ [10.2. Plagiarism](#)

Week 1 146

The following is an example of how you upload your technology abstracts to github.

[Upload Tech Abstract to github](#) ↗

Please direct any questions toward the TA's, additionally there is a README available below.

[README](#) ↗

The report will be generated on Mondays at 9:00 am est and made available by 12:00 pm est of the same day.

- https://drive.google.com/open?id=1h6_ZRmlCRIFMHG861wSyriPzn9rXxgKT

7.2.6 Cloud Accounts

Exercise 7.20 Account Creation: github.com. If you do not have a github.com account, go to github.com and apply for a <https://github.com> account. Write down your account name and remember the password. You will need the account for upcoming assignments.

Exercise 7.21 Account Creation: futuresystems.org.

If you do not have a futuresystems.org account go to <https://portal.futuresystems.org/user/register> and apply for an account. Write down your account name and remember the password. You will need the account for upcoming assignments.

Exercise 7.22 Account Creation: chameleoncloud.org.

If you do not have an account on chameleon cloud please go to <https://www.chameleoncloud.org> and apply for an account only. Do not apply for a project. Write down your account name and remember the password. You will need the account for upcoming assignments.

Exercise 7.23 Fill out the form so we can activate the accounts for you <https://goo.gl/forms/WOMdgoJoY8F6Vt9Q2> You will need the account for upcoming assignments.

7.2.7 Entry Survey

Exercise 7.24 Please fill out the following survey ASAP as it will determine some of the class material we prepare based on your feedback. The survey is really simple and can be finished in under 5 minutes.

<https://goo.gl/forms/Q04FW9eBM7eyL0Lv1>

7.2.8 Account Setup

Exercise 7.25 Github Setup due Feb, 2018

This exercise will assist you with your github account setup. Please find the README.yml file in the link below and copy it into your github repository and fill out with your information. <https://github.com/cloudmesh-community/hid-sample/blob/master/README.yml> Remove the README.md file. After your .yml file is correctly setup you now need to put your

bio in your repository, naming it like so `bio-lastname-firstname.tex` where `lastname` is your lastname and so on. Put the `.tex` file in your repository and use the latex function `subsection{lastname,firstname}` followed by the text of your bio.

7.2.9 REST

Exercise 7.26 Read the Sections Overview of Rest Section 34 and Eve 34.3.

Exercise 7.27 Develop an Eve REST Service due Feb, 2018

See: <https://piazza.com/class/jbku81aeli95rz?cid=55>

In this exercise, you will be developing an Eve REST service related to Cloud Services. We will enhance this assignment throughout the semester once we have spoken about cloud services in more detail. At this time you are expected to write a REST service that exposes information from your computer, such as, processor name, RAM, Disk. Please identify what information would be useful to have and how to obtain that information related to your operating system. Additionally identify how to integrate dynamic data.

Exercise 7.28 Assignment catch up: please look at all previous assignments and do them.

Exercise 7.29 AI cloud service: This assignment will evolve, therefore we just provide the link to it in piazza. Please visit it:

<https://piazza.com/class/jc9dcfnbi045kv?cid=34>

7.2.10 Swagger REST Services

PART A: Elementary Swagger Server

Exercise 7.30 This is for residential students (The swagger preparation was mentioned last week in class):

- (A) you are expected to have done major portions of the Swagger code gen assignment this includes
 - (1) pick of a resource you like to implement a REST service for
 - (2) read the swagger spec
 - (3) complete the spec for the resource as much as possible
 - (4) generate the code via swagger codegen

This naturally means you need swagger, python and other needed libraries set up on your machine. We will help refining your resource. We will help working with you on a simple back-end implementation that you will improve. You will be using github for all of this

- (B) In addition to this I recommend that you
 - (1) find on the network how to put Rasbian on an SD card, create your self an md file for this as this may be different for different operating systems.
 - (2) download rasbian on your machine so you can burn it as you will get 5 sd cards In the lab, those that finish most of the swagger service will switch to building a cloud cluster.

- (3) in the lab you will be first thing change the password to each of them before you put them on the network. Maybe there is a way to do this directly from your computer after you have put rasbian on the sd card. But I do not know if that's possible.
- (4) figure out how to configure Rasbian without a monitor, while just using an SDcard and your laptop, write an md file

In case of questions, lets engage in a discussion. If you have md files for information already in your repo, please post URLs. This assignment is about openly sharing. Naturally, you should not wait till someone else does it, you should take leadership yourself.

Naturally, focus on the swagger service before starting to get more involved with the cluster

Exercise 7.31 Cloud and Big Data REST Service with Swagger due March 5, 2018 Spec before Feb 15, 2018

- (A) Read the Chapters about REST and Swagger. This includes Swagger specification and Swagger codegen. There is also a video that introduces you to Swagger
 - https://youtu.be/0_Ub13py_K8
- (B) Read the Document
 - <https://laszewski.github.io/papers/NIST.SP.1500-8-draft.pdf>

We will be collaboratively developing a new version of this document while not using examples as in the previous document but swagger OpenAPI 2.0 specifications.

- (C) You will pick one of the existing resources or identify a new resource that you would like to specify. Simply go to:
 - <https://docs.google.com/document/d/12FUtH1EzQwxc3hjyki3RbU0iMfBrcyoSMKeq3aEDPk/edit?usp=sharing>
 and add your name to one of the resource. Make sure there is only one name for each resource even if you work in a team.
- (D) For the resource, you chose you will be developing with Swagger a useful REST service related to cloud computing and Big Data.

TA's will provide more details as we will avoid that everyone develops the same service.

You will use this specification create a swagger python service and implement functions to enable a real implementation of the service that is useful.

Examples:

- (A) develop a service to upload files to a file system with REST calls.
- (B) start a cluster of virtual machines on a supercomputer
- (C) start a cluster of virtual machines on OpenStack

Additional resources are listed in the instructor answer.

PART B: Reproducible Swagger Services

Exercise 7.32 It is important to be able to reproduce the Services and not just create a code that you can run. In order to do that we will only store the absolute minimum information in github and autogenerate the code via the yaml file, your controller and a Makefile that you will design.

Naturally we needed you to have understood the Swagger codegen tool first, so you need to be familiar on how to create a swagger service. Now that you are we can continue with this step. This will even include removing code that you uploaded to github.

Thus, We like you to review the following and engage with us in online meetings if this is not clear. It is actually rather simple. This is used to prepare you for the way we expect you to deliver the final project. It also replicates the way we have taught you to compile your latex pdf document. Wait -- what has latex to do with swagger services? Technically nothing conceptual we do

- make in paper dir -> produces latex document
- make in swagger dir-> produces swagger service
- and in future make in project dir -> produces project services

So we just use the same framework, which is very convenient!

Now let us get back to the swagger service generation:

One of the goals of this project is to create a REPRODUCABLE framework for generating the services. It is not enough to just develop a swagger service. You will need to generate the service from a shell script and/or makefile. As both technologies are available on any computer including Windows it is your responsibility to make sure you have make and bash installed. And use them.

Swagger-codegen applied to your yaml file will create a directory structure. This directory structure is not to be checked into github. Instead you will check in the makefile or shell script (bash) that creates it.

You will see in the controller dir a number of `controller_*.py` files, you will copy them into your github dir. The names of the controller files can be automatically specified based on the content in your yaml file.

Thus you will have a directory with the following contents, replace 000 with the id you have

```
1 hid-sp18-000/swagger/Makefile
2 hid-sp18-000/swagger/controller_a.py
3 hid-sp18-000/swagger/controller_b.py
```

We assume that `swagger-codegen` is a shell variable allowing you to run `swagger-codegen`.

This could be different on different systems. You will be documenting this for your system. On OSX this is trivial as you just use brew to install and

```
1 export SWAGGER-CODEGEN=swagger-codegen
```

will typically work. On other systems you may have to specify the jar file. You will be using an environment variable regardless which OS you are on

If you use a Makefile you will be defining the following tags

- make clean -- removes the code generated
- make service -- creates the swagger service from the yaml file and places the controllers in the appropriate directory
- make start -- starts the service
- make stop -- stops the service
- make test -- executes a number of tests against the service

You are allowed as part of this use nose or any other unit test.

Swagger Container

Exercise 7.33 STEP 3 generating Swagger REST Containers

After you have finished the STEP 2 in the Swagger series of exercises, You will now generate a container. However, you will neither check in the container in docker hub nor into github.

Instead, you will generate a container with a Dockerfile. You will add to your makefile the tag

```
1 make container -- which will generate the container form the
2                 Dockerfile. Your container will be named cloudmesh-<↔
   ↪ YOURTOPIC>
```

Where topic is how you named your service

As we will start multiple services from multiple students you need to have a proper namespace in the yaml specification file This may be different for different people. For example

```
1 cloudmesh/var/<id>
2 cloudmesh/aws/vm/<id>
3 cloudmesh/google/vm/<id>
4 cloudmesh/openstack/vm/<id>
5 cloudmesh/filter/<id>
```

Naturally, if you have better url suggestions please integrate, however, we need a unique prefix for each service so if we were to combine them we can do that.

Reply to this followup discussion ■

7.2.11 Technology Paper**Exercise 7.34 Technology Paper** due March 20, 2018

Read the following two points to assist you in starting your paper

- (A) pick one of the technologies identified by you or if you see a hid that has already picked a technology for this assignment you can also pick one form that students list also
- (B) write a LaTeX paper about the technology. Make sure not to plagiarize. The maximum number of quotes is about 25% if needed. Please see the scientific writing section

WE RECOMMEND YOU GET STARTED ON THIS RIGHT AWAY AS YOU ALSO WILL HAVE TO DO A TUTORIAL AND THE FINAL PROJECT.

Additional details can be found here: <https://github.com/cloudmesh-community/hid-sample/blob/master/paper-instructions.md> ■

7.2.12 Tutorial**Exercise 7.35 Tutorial** due March 26, 2018

This is for online and residential students.

Residential students with a substantial tutorial such as install a kubernetes cluster on PI or docker swarm or some other larger topic are exempt from this assignment. However, if you are working towards an A+ consider adding an additional tutorial.

We like you to pick a technology and develop for this technology a tutorial. The tutorial can be written either in markdown or in LaTeX. However, when we like that you do not use enumerations for steps that you document. Instead use sentences such as

First, we do

Second, we need to

Next, we implement

Please use sections, subsections and so on to the structure. Additionally please create a separate directory for images called **images** where all the images are stored like the structure below.

```
1 hid-sp18-000/tutorial/images
2 hid-sp18-000/tutorial/tutorial.tex
```

DO NOT USE SCREENSHOTS FOR CODE EXAMPLES OR CAPTURE OF THE TERMINAL OUTPUT. USE ASCII and put it either in `lstlisting` in latex, or in an indented codeblock in markdown. Please do not add python, bash or other markings to your codeblock, as we need real simple markdown if you chose that. Make sure text in LaTeX.

We will make additional suggestions for tutorial topics in the document in which we collect the tutorial list. We will also indicate for these suggesting a maximum number of students able to work together on that tutorial.

[Tutorial List→](#)

7.2.13 Project

Exercise 7.36 The policies and format about the project are discussed in Sections 8.15 and 8.15.3.

Some project idease for this years class are listed in Section 8.16. Some previous projects are listed in previous volumes that you can find in Section 1.

It is purpose of this class that you define your own project first. There should not be an overlap between other projects. All projects should be approved. Two month before the project deadline the class was informed via Piazza to think about the project and engage in a discussion about the topic. A one page snapshot of the project paper is due on April 2nd as posted in piazza.

The final paper is due Fri 4/27 9am. This is a hard deadline.



Class Policies and Communication

8	Course Policies	115
8.1	Communication and Use of CANVAS	
8.2	Managing Your Own Calendar	
8.3	Online and Office Hours	
8.4	Class Material	
8.5	HID	
8.6	Notebook	
8.7	Blog	
8.8	Calendar I524, E616, E516	
8.9	Incomplete	
8.10	Waitlist	
8.11	Registration	
8.12	Auditing the class	
8.13	Resource restrictions	
8.14	Plagiarism	
8.15	Tutorials, Topic Paper, Term Paper, Project Report	
8.16	Project Ideas	
8.17	Grading	
9	Piazza	133
9.1	Access to Piazza from Canvas	
9.2	Verify you are on Piazza via a post	
9.3	Making Piazza Work	
9.4	Towards good questions	
9.5	Guide on how to ask good questions	
9.6	Piazza class Links	
9.7	Piazza Curation	
9.8	Read the Originals, not just the e-mail	
9.9	Exercises	
9.10	Fall 2018	



8. Course Policies

We describe briefly how we manage different classes and how to interact with us.

8.1 Communication and Use of CANVAS

In the past we have found numerous limitations to CANVAS that makes it not possible for us to use it effectively for our classes. Such limitations were not overcome even after consulting our professional course support team at IU although we spend days and weeks trying to fix the issues we encountered. From the professional staff we have been given the advice to avoid using CANVAS and use a better approach that we outline as follows:

- we use CANVAS notifications only for the initial class setup and notify you how to gain access to piazza where all class discussions take place;
- any discussion about grades is however only done in CANVAS;
- we will DELETE all e-mail sent to us on personal or IU e-mail addresses. All discussions need to take place either in private or in public posts on piazza (other than grades which need to be sent via CANVAS as the previous point explained);
- TA's are strictly forbidden to answer any e-mail that is not sent via piazza. Their answers will only be posted within piazza;
- any post to piazza to a TA must be not done to the individual TA but to *instructors*;
- any post from piazza by a TA must be not done to the individual student only, but also must include *instructors*;

8.2 Managing Your Own Calendar

From time to time we get the question from a very small number of students why we are not using or uploading the assignment deadlines and the assignment descriptions to CANVAS. The reason

for this is manifold. First, our class has different dealines for different students within the same class. This is not supported by CANVAS and if we would use CANVAS leads to confusion and clearly shows the limitation of CANVAS. Second, we are teaching cloud computing. CANVAS is not a tool that you likely will use after graduating. Thus we are providing you the ability to explore industry standard tools such as github to maintaining your own tasks and deadlines, while for example using github issues (see the section about github). We highly recommend that you explore this as part of this class and you will see that managing the assignments in github is **superior** to CANVAS. Naturally you can not make that assessment if you are not trying it. Thus we like you to do so and it is part of any assignment in your class to use github issues to manage your assignments for this class.

However, if you still want to manage your tasks in CANVAS, you can do so. CANVAS allows you to create custom events, so if you see an assignment in piazza or the handbook, you are more than welcome to add that task yourself to your own CANVAS tasks. As we have only a very small number of assignments this will not pase a problem either for graduate or undergraduate. Being able to organize your deadlines and assignment with industry accepted tools is part of your general learning experience at IU.

Obviously, this makes it also possible to use any other task or calendar system that you may use such as google calendar, jira, microsoft project, and others.

As you can see through this strategy we provide the most flexible system for any student of the class, while giving each student the ability to chose the system they prefer for managing their assignment deadlines. It is obvious that this strategy is superior to CANVAS as it is much more general.

8.3 Online and Office Hours

To support you we have established an open policy of sharing information not only as part of the class material, but also as part of how we conduct support. We establish the following principals:

- in case of doubt how to communicate address this early in class and attend online hours;
- all office hours if not of personal nature are open office hours meaning that any student in class can be joined by other students of the class and all meeting times are posted publicly.
- it is in your responsibility to attend in person classes and online hours as we found that those that do get better grades. For residential students participation in the residential classes is mandatory due to the same reason.
- instructors of this class will attempt within reason to find suitable times for you to attend an online hour in case you are an online student. Residential students can attend the class on Friday and ask any question.

8.3.1 Office Hour Calendar

Online Students: Online hours are prioritized for online students, residential students should attend the residential meetings.

Residential Students: Residential students participate in the official meeting times. If additional times are required, they have to be done by appointment. Office hours will be announced publicly. All technical office hours are public and can be attended by any student.

Online hours are not an excuse not to come to the residential class.

However Residential students can in addition to the residential class use the online student meeting times. However, in that case online students will be served first. It is probably good

to check into the zoom meeting and identify if the TA has time. They will be in zoom.

We suggest that you let the TA's know in piazza before you come, in order to make sure they are at the office. The online support hours are as follows:

- Mon 6:00pm-7:00pm, 7:00pm-8:00pm, Gregor
- Tue 7:00pm-8:00pm, Tyler, Vibhatha, Hyungro, Bo or Miao
- Wed 7:00pm-8:00pm Bo
- Thu 8:00pm-9:00pm Hyungro
- Fri 7:00pm-8:00pm Miao
- Sat 10:00am-11:00am Vibhatha
- Sun 10:00am-11:00am Tyler

If a meeting is needed with Gregor, this is done upon appointment Tue-Thu 10am - 2:30pm. However, TA's will figure out if a meeting is needed. Please prepare your technical questions ahead of time, and place them in Piazza first. TA's and the class will try to answer them if possible

The link for joining the meeting on Zoom is posted in Piazza.

- [TBD](#)

For more up-to-date details, refer to Piazza.

8.4 Class Material

Warning

As the class material will evolve during the semester it is obvious that some content will be improved and material will be added. This benefits all classes. To stay up to date, please, revisit this document on weekly basis. This is obvious, as we will adapt content based on your feedback.

The requirement of the classes to become an expert in cloud and/or Big Data applications and technologies stays unchanged.

8.5 HID

You will be assigned an hid (Homework IDentifier) which allows us to easily communicate with you and do not allow us to not use your university ID to communicate with you.

You will receive the HID within the first week of the semester by the TA's.

8.6 Notebook

All students are required to maintain a *class notebook* in github in which they summarize their weekly activities for this course in bullet form. This includes a self maintained list of which lecture material they viewed.

The notebook is maintained in the class github.com in your hid project folder. It is a file called notebook.md that uses markdown as format. While using md, you can either edit it locally and upload to github, or directly edit it via the git hub Web editor. Notebooks are expected to be set up as soon as the git repository was created.

You will be responsible to set up and maintaining the notebook.md and update it accordingly. We

suggest that you prepare sections such as: Logistic, Theory, Practice, Writing and put in bullet form what you have done into these sections during the week. We can see from the github logs when you changed the notebook.md file to monitor progress. The management of the notebook will be part of your discussion grade.

The format of the notebook is very simple markdown format and must follow these rules:

- use headings with the # character and have a space after the #
- use bullets in each topic.
- each bullet **must** have an individual date that is of the form yyyy/mm/dd. Please do not lump bullet points under a single date. Have each bullet point its own date
- if you have done the activity in a period then add the second date to it yyyy/mm/dd - yyyy/mm/dd
- If you refer to section numbers in your notebook, please also add the section title as the section numbers may change in case we need to add content

Please examine carefully the sample note book is available at:

- <https://raw.githubusercontent.com/bigdata-i523/sample-hid000/master/notebook.md>

This will render in git as:

- <https://github.com/bigdata-i523/sample-hid000/blob/master/notebook.md>

The notebook.md is not a blog and should only contain a summary of what you have done.

8.7 Blog

Naturally you can maintain your own blog, but it is optional. If you like to maintain your own blog, you can create yourself also a blog.md file. However do not include sensitive information in there. A blog is not a replacement for the notebook. The blog will not be used for grading. If something does not go so well, do not focus on the negative things, but focus on how that experience can be overcome and how you turn it to a positive experience.

8.8 Calendar I524, E616, E516

This class is a full term class of 16 weeks.

Indiana University

The semester calendar is posted at

- <http://registrar.indiana.edu/official-calendar/official-calendar-spring.shtml>

The class begins Mon, Jan 8th and ends Fri, May 4th

8.9 Incomplete

Incompletes have to be explicitly requested in piazza through a private mail to *instructors*. All incompletes have to be filed by May 1st.

Incomplete's will receive a fractional Grade reduction: A will become A-, A- will become B+, and so forth. There is enough time in the course to complete all assignments without getting an incomplete.

Why do we have such a policy? As we teach state-of-the-art software this software is subject to

change, not only within the course, but also after the course. As we may offer some services and only have access to the TA's during the semester it is obvious that we like all class projects and homework assignments to be completed within a semester. Services that were offered during the semester may no longer be available after the semester is over and could adversely effect your planing. It will be in the students responsibility to identify such services and provide alternatives if they become unavailable. We try hard to avoid this but we can not guarantee it.

Furthermore, once an incomplete is requested, you will have 10 month to complete it. We will need 2 month to grade. No grading will be conducted over breaks. This may effect those that require student loans. Please plan ahead.

The incomplete request needs to be off the following format in piazza:

```
Subject:
  INCOMPLETE REQUEST: HID000: Lastname, Firstname

Body:
  Firstname: TBD
  Lastname: TBD
  HID: TBD
  Semester: TBD
  Course: TBD
  Online: yes/no

  URL notebook: TBD
  URL assignment1:
  URL assignment2: TBD
  ....
  URL paper1: TBD
  URL project: TBD

  URL other1: TBD
```

Please make sure that the links are clickable in piazza. Also as classes have different assignments, make sure to include whatever is relevant for that class and add the appropriate artifacts.

Indiana University

Here is the process for how to deal with incomplets at IU are documented:

- <http://registrar.indiana.edu/grades/grade-values/grade-of-incomplete.shtml>

Piazza is FERPA compliant:

- <https://piazza.com/legal/ferpa>

8.10 Waitlist

The waitlist contains students that are unable to enroll in a section of a course. Students choose to add themselves to the waitlist. They are not automatically added, but choose to do so intentionally based on the status of the course. There are two reasons for students to be on the waitlist. The first, and primary, reason is that the class is already at the scheduled, maximum capacity. Since there are no seats available, the student can elect to add themselves to the waitlist. The second reason is that the students' own schedule has a time conflict. This occurs when they are trying to enroll in a class that overlaps with the time of a class they are already enrolled in.

Students are moved from the waitlist to the regular section during a daily batch process, and not in real time. The process is not in realtime because the registrar receives many requests to increase capacity, decrease capacity, and change rooms. If the process were real time there would be a

catastrophe of conflicts.

Students are moved from the waitlist in chronological order that they added themselves to the waitlist. If you are still on the waitlist there are no spaces free, the batch process has not run for the day, or the student in question has a schedule conflict.

Faculty are not able to selectively choose students from the waitlist.

How long does the waitlist process stay active?: The automated processing of the waitlist ends on Thursday of the first week of class. At this time the waitlist will no longer be processed. As the residential class starts on Friday, this may cause issues. Either talk to the department on Thursday or show up on Friday. Most likely there will be spaces left. Students on the waitlist at that time will remain on the waitlist, but remain there until the student decides to change their registration. Students may not do that, because they get assessed a change schedule fee.

Students tell me they still want to enroll after the first week of classes. How do they do this?

Beginning Monday, after the first week of class students begin to use the eAdd process to do a late addition of the course. The request is routed to the professor of record on an eDoc and the faculty will be notified via email. Faculty can deny or approve based on whatever criteria they wish to apply. If the faculty member approves, the eDoc is electronically forwarded to the Academic Operations office and we will approve the late add **if the room capacity** allows the addition, otherwise we must deny the addition because of fire marshal regulations. Many times, there are seats in a classroom/discussion/lab, but because other students have not *officially* dropped, enrollment is still at capacity.

After everything, a student that was unable to enroll in the class attended all year and completed all course work as if they had enrolled. Can the student get credit and can I give the student a grade?

Yes. There is a provision for a late registration - contact our office if this occurs. Students will be assessed a tuition fee at the time of late or retroactive registration.

8.11 Registration

The Executive Associate Dean for Academic Affairs requires starting Spring 2018 that students that are not officially enrolled, can not register at the end of the class if they in-officially took the class. Please make sure that within the first month you have enrolled. If we do not see in CANVAS, you are not in the class. In case you are on a waitlist it is your responsibility to work with the administration after the waitlist is over to be added to the class by getting permission from the School.

8.12 Auditing the class

We no longer allow students to audit E222, I524, E516, and E616. The motivation to not offer these classes for auditing are:

- Seating in the lecture room is limited and we want foster students that enroll full time first.
- The best way to take the class is to conduct a project. As this can not be achieved without taking the class full time and as auditing the class does not provide the full value of the class, e.g. not more than 10% of the class. Hence, we do not think it is useful to audit the class.
- Accounts and services have to be set up and require considerable resources that are not accessible to students that audit the class.

8.13 Resource restrictions

- It is not allowed to use our services for profit (e.g. just enrolling in the class to use our clouds).
- In case of abuse of available compute time on our clouds the student is aware that we will terminate the computer account on our clouds and the student may have to conduct the project on a public cloud or his own computer under own cost. There will be no guarantee that cloud services we offer will be available after the semester is over. Projects can be conducted as part of the class that do not require access to the cloud.

8.14 Plagiarism

In the first week of class you will need to read the information about plagiarism. If there are any questions about plagiarism we require you to take a course offered from the IU educational department.

Warning

If we find cheating or plagiarism, your assignment will be receiving an *F*. This especially includes copying text without proper attribution. In addition you will be receiving an *F* for the appropriate time for the discussion points an assignment was issued, e.g. If a paper duration assignment is 4 weeks, you get for these four weeks no discussion points, meaning an *F*. Furthermore, we will follow IU policy and report your case to the dean of students who may elect to expell you form the university. Please understand that it is your doing and the instructors have no choice as to follow university policies. Thus, please do not blame the instructors for your actions. Excuses such as “I missed the lecture on plagiarism”, “I forgot to include the original reference as I ran out of time”, “I did not understand what plagiarism is” do not count obviously as we explicitly make the policies clear. This applies to all material prepared for class including assignments, exercises, code, tutorials, papers, and projects. If there is no time, do not submit and instead of an *F* ask for an incomplete. In fact if you know you have plagiarized, do not even have us review your paper.

For more information on this topic please see:

- <https://studentaffairs.indiana.edu/student-conduct/misconduct-charges/academic-misconduct.shtml>

8.15 Tutorials, Topic Paper, Term Paper, Project Report

Dependent on the class you need to do different assignments. The assignments will be clearly posted in this document and updated in case clarification is needed.

We use the following terminology:

Tutorials: Tutorials are written in markdown, RST, or LaTeX and include information on a particular technical issue that is in general helpful for other students. Tutorials can be small, but some may need to be substantial. As we expect that the tutorials can be included in the Handbook, please be careful of plagiarism and do not just copy the tutorial from elsewhere.

Topic Paper: A topic paper, or short paper is a snapp paper about a technology, application, or useful information that provides an overview of what you are trying to describe and analyses its relationship to the class topic. Be mindful about plagiarism. The paper is written in LaTeX and uses jabref for bibliography management.

Term Paper: A term paper is an enhanced topic paper. The difference is in length. Comparative or review papers can also be term papers. Term papers should have the quality to be publishable either in a workshop or as part of the handbook.

Project Paper: A project report is an enhanced topic paper that includes not just the analysis of a topic, but an actual code, with benchmark or demonstrated application use. Obviously it is longer than a paper and includes descriptions about reproducibility of the application. Term papers should have the quality to be publishable either in a workshop or as part of the handbook.

Assignments: In addition to the previously discussed topics you also are doing a small number of assignments. These assignments may take you one or multiple weeks to accomplish. Some of them are pass fail, while others will receive a grade. It will be clearly stated at the beginning of the assignment which of the evaluation will apply.

Examples from prior classes are available in the class proceedings listed in Section 1.

Dependent on the class you have to fulfill different requirements. Please make sure you understand which requirement you will have.

E516 In these classes you will need to produce tutorials, topic papers and a project report with real code.

E616 In these classes you will need to produce tutorials, topic papers and a project report with real code.

I524 same as E616, but you have the choice to substitute the project report with a term paper.

Please be aware that the project or term paper constitute to a significant portion of your grade of your class grade. You have plenty of time to make this choice and if you find you struggle with programming you may want to consider a term paper instead of a project.

In case you chose a project your maximum grade for the entire class could be an A+. However, an A+ project must be truly outstanding and include an exceptional project report. Such a project and report will have the potential quality of being able to be published in a conference.

In case you chose a term Paper for I524 your maximum grade for the *entire* class will be an A-.

Please note that a project includes writing a project paper. However the length is a bit shorter than for a term paper.

8.15.1 Team

Software projects and term papers can be conducted with one, two or three class members. We do not allow more than three members in a project, paper, or assignment team. It will be up to you to determine a team, but we recommend that you choose wisely. Naturally if a team member does not contribute to the project you need to address this early on. Please do not come to us a week before the deadline is due and say a team member has not contributed, this is far too late to do any adjustment to the team. It is in your responsibility to manage the team. You can build different teams throughout the semester for different tasks. Please communicate clearly and timely with your class mates.

8.15.2 Common Deliverables

Both Projects and Term paper have the following common deliverables

Work Breakdown: This is an appendix to the document that describes in detail who did what in the project. This section comes in a new page after the references. It does not count towards the page length of the document. It also includes explicit URLs to the the git history that documents the statistics to demonstrate not only one student has worked on the project. If you can not provide such a statistic or all checkins have been made by a single student, the project has shown that they have not properly used git. Thus points will be deducted from the project. Furthermore, if we detect that a student has not contributed to a project we may invite the student to give a detailed presentation of the project.

Bibliography: All bibliography has to be provided in a jabref/bibtex file. This is regardless if you use LaTeX or Word. There is **NO EXCEPTION** to this rule. Please be advised doing references right takes some time so you want to do this early. Please note that exports of Endnote or other bibliography management tools do not lead to properly formatted bibtex files, despite they claiming to do so. You will have to clean them up and we recommend to do it the other way around. Manage your bibliography with jabref, and if you like to use it import them to endnote or other tools. Naturally you may have to do some cleanup to. If you use LaTeX and jabref, you have naturally much less work to do. What you choose is up to you.

Report Format: All reports will be using the our common format. This format is not the same as the ACM format, so if you use systems such as overleaf or sharelatex, you need to upload it and use it there.

The format for LaTeX and Word found here:

- <https://github.com/cloudmesh-community/hid-sample/tree/master/paper>

There will be **NO EXCEPTION** to this format. In case you are in a team, you can use either github while collaboratively developing the LaTeX document or use Microsoft One Drive which allows collaborative editing features. All bibliographical entries must be put into a bibliography manager such as jabref, or Mendeley and exported to bibtex. This will guarantee that you follow proper citation styles. You can use either ACM or IEEE reference styles. Your final submission will include the bibliography file as a separate document.

Documents that do not follow the ACM format and are not accompanied by references managed with jabref or are not spell checked will be returned without review.

8.15.3 Project Paper

Systems Usage

Projects may be executed on your local computer, a cloud or other resources you may have access to. This may include:

- chameleoncloud.org
- furturesystems.org
- AWS (you will be responsible for charges)
- Azure (you will be responsible for charges)
- virtualbox if you have a powerful computer and like to prototype
- other clouds, please confirm with us.

Access to clouds must be scripted and a cmd5 extension must be developed as part of your project to receive full credit.

Deliverables

The following artifacts are part of the deliverables for a project

Code: You must deliver the **source code** in github. The code must be compilable and a TA may try to replicate to run your code. You **MUST** avoid lengthy install descriptions and everything must be installable from the command line. We will check submission. All team members must be responsible for one or all parts of the project.

Code repositories are for code, if you have additional libraries that are needed you need to develop a script or use a DevOps framework to install such software. Thus zip files and .class, .o files are not permissible in the project. Each project must be reproducible with a simple script. An example is:

```
git clone ....
make install
make run
make view
```

Which would use a simple make file to install, run, and view the results. You are expected to integrate cmd5, which we teach in class. In addition you can use or are expected to use DOCKERFILES, ansible, or shell scripts. It is not permissible to use GUI based DevOps preinstalled frameworks. Everything must be installable and reproducible form the command line.

Data: Data is to be hosted on IUs google drive if needed. If you have larger data, it should be downloaded from the internet. It is in your responsibility to develop a download program. The data **must** not be stored in github. You will be expected to write a python program that downloads the data.

Project Report: A report must be produced while using the format discussed in the Report Format section. The following length is required:

- 6 pages, one student in the project
- 8 pages, two students in the project
- 10 pages, three students in the project

License: All projects are developed under an open source license such as Apache 2.0 License. You will be required to add a LICENCE.txt file and if you use other software identify how it can be reused in your project. If your project uses different licenses, please add in a README.md file which packages are used and which license these packages have.

8.15.4 Term Paper

In case you chose the term paper, you or your team will pick a topic relevant for the class. You will write a high quality scholarly paper about this topic. The following artifacts are part of the deliverables for a term paper. A report must be produced while using the format discussed in the Report Format section. The following length is required:

- 8 pages, one student in the project
- 10 pages, two student in the project
- 12 pages, three student in the project

8.16 Project Ideas

For the format and the details about artifacts produced in general for projects, please see Section 8.15. This includes the length and the paper format.

The paper format is included in hid-sample project-report (e.g. same as paper):

- <https://github.com/cloudmesh-community/hid-sample/tree/master/project-paper>

The code and the paper are to be added in your hid folder. YOU will be creating lower case directories called project-paper and project-code. You will not check in any data, but instead create scripts that fetch the data.

For an example directory structure, please see

- <https://github.com/cloudmesh-community/hid-sample>

Certainly, you can chose from many different topics and we hope you pick one that is suitable for you and you enjoy doing. You have the opportunity to definitely pick a project that you enjoy doing. However it must be related to the course. This course is not about finding the best algorithm or copying a project from your AI or other cloud classes you have taken at IU. It is about finding a novel project that is related to cloud computing, big data and the deployment of the system on cloud resources.

8.16.1 Project Data Restrictions

On Mar 8th we posted in Piazza, that there are a couple of restrictions to the choice of the data set. We will not accept any project using the Titanic, Wordcount, or any Kaggle dataset.

8.16.2 Example Projects

Example projects are available in the Volumes we published and are listed in Section 1.

8.16.3 Register your project idea

Please register your project idea here:

- https://docs.google.com/document/d/14L0guBfWJdRqqf1BoLw41LPg9HU3BqR0FZeF43hJF_E

8.16.4 Meeting with Gregor

On March 11th we posted that you need to meet with Gregor to discuss your projects. Many students have done this, but we are missing some that have not yet started or postponed the idea of starting the project. Please attend Monday March 25, sometime between 7pm to 10pm. If you can not attend this meeting, please identify an alternative time where we will discuss your project in a group setting. Have your project idea ready so we can discuss it.

8.16.5 Project type A: NIST Rest services project

This project idea is the simplest one of the once listed in this section as we have extensively discussed it and provided all important information to succeed in this activity. However, this task

must not be underestimated as it requires some non trivial work as any of our other tasks. We believe however that the efforts for it is smaller than with other project ideas.

We have provides you with the NIST big data reference architecture. As part of this we have identified how to create rest services. In this project you will define a **SIGNIFICANT** set of resources and implement the rest services for them. IN contrast to your previous assignment this is a set of services and not just a single service. (for example just implementing a key value store abstraction is not sufficient). A proper project scope includes for example an abstraction of resources related to VMs on AWS with libcloud, or VMs on Openstack with libcloud, or VMs on Azure with libcloud, or An abstraction for data storage (not just files, but also objects and key value pairs), the abstraction of an accounting framework, and so forth.

In case you have not completed your swagger REST service a portion of this project will be used to satisfy that requirement.

8.16.6 Project type B: Raspberry Pi projects

The raspberry Pi projects are divided topically by class. While 516 focusses on map reduce 616 and 524 are focussing on containers. Exceptions could be allowed with proper reasoning.

e516

In this project you will be developing or leveraging form an existing tutorial developed as part of the class. You will be focussing on how to create a Spark and/or Hadoop cluster for Raspberry Pi's in a scalable fashion. In previous tutorials for the class students focussed on setting this up for a small number of nodes. What we need to do now is to expand this to a scalable solution with many hundrets of Pi's in the cluster. Naturally login in by hand on these machines is not suitable, but you need to automatize this process as much as possible. Your ideas on how to do this are most welcome. There are different strategies, such as burning all SD cards with a program on your laptop and modifying the file system of the sd card after the burning, setting up a simple minimal system with ssh enabled and DHCP so you can log into a named host and use parallel commands to further provision the system, or even PXE boot. Once you have figured out and documented this you will be documented how to deploy a Hadoop and/or a spark cluster on the Pis.

You will then pick a data set and do a mapreduce application and measure the performance.

In case you work in a team, each person in the team needs to add a new deployment. Example, if you are in a 3 person team you need to do not only do a single deployment but multiple. This could even mean that you need to deploy it on echo which is a non Pi cluster, but you can get great performance comparisons between your analysis on echo and the one on the PI. Other examples could include the comparison of spark with hadoop on PI and echo

As this project contains a significant of tutorial like activities (just do not use the term tutorial, but in this section we describe) we recommend that you develop the setup procedure in markdown. and not directly in latex. However use *clean* markdown and follow the markdown rules. We have seen in the tutorial to be delivered for this class many wrong examples on how to not use markdown.

For this reason the length of the paper may be reduced by one page if the set up procedure is excellent, and includes automated deployment scripts with minimal input by hand (this requires programming).

e616 and I524

In e616 and I524 we have essentially the same requirements as in Section 8.16.6, but replace map, reduce, hadoop, and spark with containers, Docker Swarm and Kubernetes. Please remember here you will have a cluster with docker swarm or kubernetes, It is not sufficient to just install docker on all nodes.

All other requirements are the same as in Section 8.16.6.

8.16.7 Project type C: Data related project for Spark or Hadoop

This project requires you to use one cloud IaaS resource such as chameleon, Futuresystems Echo, AWS, or Azure.

You will be deploying on the IaaS a Spark or Hadoop cluster and conducting based on a data set that you conduct an analysis of the data. YOU will be benchmarking the time it costs to set up this environment as well as benchmarking how fast the analysis is.

8.16.8 Project type D: Data related project for a kubernetes or swarm cluster

This task is not for e516 students.

This project requires you to use one cloud IaaS resource such as chameleon, Futuresystems Echo, AWS, or Azure.

You will be deploying a multi nod kuberntes or swarm cluster (in case of echo this is already done, so you will just use it, but substitute the deployment task with somethings else).

You will be deploying rest services on the cluster base don the Swagger services we explained to you and benchmarking them. The rest services are related to the Big data architecture or have significant set of analysis components.

In case you run on echo, you just do more services than you would do if you were running on other platforms.

One nice project would fo =r example be the automated creation of rest services while using a function specification of python. THis way you could for example look at scikit learn, write 10 use cases, use your code generator and create for each of them the rest service. Important would be a scalability test.

8.16.9 Project type E: Define your own

Define your own project and discuss with us in the Monday meeting with Gregor. A good example is a student that has chosen grapQL as the major infrastructure component. He is developing a contributed chapter for the handbook, a tutorial, and a deployment and benchmark of data of his choice.

8.16.10 Project Idea Piazza Notes

In addition to the above notes we have selected some postes from piazza in which we discussed project related activities. As they are from different classes, we posted the content and not just the URL.

A project idea to create a spark kubernetes cluster

On March 11 we posted in 516:

- <https://piazza.com/class/jbku81aeli95rz?cid=274>

Tutorial and Project Idea: Reproducible Scalable spark cluster and benchmarking. While on the phone with others, I was asked is there a tutorial about spark that would get 10 points.

Here is an example of such a tutorial:

<http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-1/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-2/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-3/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-4/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-5/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-6/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-7/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-8/> <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-9/>

This could naturally be the basis of your project. However you can not just paste and copy, you need to write it as a section and not use the word tutorial. You need to make meaningful modifications or enhancements to it. Such as creating a Dockerfile doing all of this in an elegant fashion without any human input other than starting the process. We know this is possible and can be done. Then you need a dataset and test your deployment on a variety of machines.

3 committed people can work on this.

8.16.11 Docker Cluster on PI Video

The following video is pretty interesting as it shows many of the steps that are needed to create a docker cluster. This is regardless if you use a cluster based on zeros, 3 B, 3B+.

- <https://www.youtube.com/watch?v=qSpfWP-FgjC>

Naturally, the video shows how to do things by hand. To bring this to the next level, One could, for example, provide a host file with the static addresses (or create them) and use it as part of a script to modify Vanilla SD cars that contain the vanilla OS on it.

E.g. the tutorial contains many steps that ask to manipulate things by hand. This is unnecessary as all the steps can be provided by a script.

The reason we want everything scripted is that we like to replicate this many many times as we want to replicate a swarm cluster for example on 100 PIs doing this on 1-4 by hand may be reasonable, but doing this on 100, we have to further automate this.

Using just 4 zeros is a good way to test this automated setup.

This could become a project. Then you just develop some swagger rest services and try to place them on the swarm. Similar things can be done with kubernetes.

8.16.12 Hadoop 3.0

Indiana University

Please note that TAs may already have done this and if so it may no longer be eligible. However, it could be extended upon.

Tutorial and Project Idea: Hadoop on Docker with newer version (a) use ubuntu image, (b) use

newest version of hadoop, (c) develop docker file similar to <https://github.com/sequenceiq/hadoop-docker/blob/master/Dockerfile> or others you can find, (d) find dataset and benchmark on various machines.

Max 3 people can work on this, while deploying it on 3 platforms and showcasing it works with a benchmark.

Project Cloud Security

On February Feb 27 we posted

Before engaging in discussions about this with me read up on security and attribute base security. Take a look at the already developed Web services to showcase how we develop flask and swagger servers with basic auth (needed to understand the attribute based security).

- https://en.wikipedia.org/wiki/Attribute-based_access_control

This project has three parts and could be used throughout the class for all assignments.

Paper. Survey of Attribute based security and other security for clouds pages = number of people * 2 maximal 3 people (no images as usual in page number)

Tutorial. find frameworks in Python that do this. If they exist to develop a tutorial

Swagger. develop a swagger rest service managing the attributes and entities in the framework

Alternative A: Project VM based. : showcase this in a project that does this in a cloud framework using distributed virtual machines and services. Develop a tool that autogenerates services based on a function definition while also adding attribute based security.

Alternative B. do the same project but instead of using VMs do it in containers.

This project has enormous potential as (a) NIST Is highly interested in this. Publication potential of one or two papers. (b) security is hot, and (c) cloud is hot

8.17 Grading

Grading for homework will be done within reasonable time of the submission if the submission was on time. This however could still take multiple weeks. Students that miss the deadline will be graded on best effort, which could mean at the end of the semester as TAs will grade first papers that have been handed in on time. A 10% grade reduction will be given for residential students if the project is late. Some homework can not be delivered late (which will be clearly marked and 0 points will be given if late; these are mostly related to setting up your account and communicating to us your account names.)

It is the student's responsibility to upload submissions well ahead of the deadline to avoid last minute problems with network connectivity, browser crashes, cloud issues, etc. It is a very good idea to make early submissions and then upload updates as the deadline approaches; we will grade the last submission received before the deadline.

Note that the term paper or project paper will take a considerable amount of time and doing proper time management is a must for this class. Avoid starting your project late. Procrastination does not pay off. Starting a paper a day or even in the week before the deadline will allow you not to achieve your best. Late Projects or term papers will receive a 10% grade reduction.

For E516 I524, E616 the grading scene is discussed in the syllabus.

8.17.1 Grades on Canvas

The final grade for your class is **NOT** accurately posted in CANVAS. Please visit the registrar for your final grade. We have run many times into issues with CANVAS thus we try to stay as much as possible away from it. We know that the total grade will not be accurately reported in CANVAS.

Furthermore we are using only a letter grading scheme that distinguishes qualitatively between grades. Typically we do not engage in arguing if you get a point more or less. Instead we look at the artifact and decide if it is an A or A- and so on. CANVAS on the other hand uses a point system that may provide a misleading information as we do not use points.

Once the due date is passed all incomplete assignments will appear as an *F* in CANVAS. Once we receive and review the submission this grade will be changed. Please, do not contact us if you submitted late and you see an *F* in CANVAS temporarily.

8.17.2 Discussion about Grades

Should it be necessary a discussion about a grade must not be taking place via e-mail nor via piazza. YOU must use the CANVAS message feature as we want to make sure that by accident you post information to others that you do not intend to. FOR this reason we will not read such messages on pizza and in our e-mail and deleted them without reading.



9. Piazza

We use Piazza (<https://piazza.com>) because questions and answers on Piazza are community-edited and provides the opportunity not only for instructors, but also for students to contribute. Each question has a single answer edited by the students of the class and if needed an instructors' answer that is collaboratively edited by the instructors.

Due to this wiki-style Q&A, when a student has a question, one does not have to look through long e-mail threads but instead can look at the answer. For details that lead up to the answer you are highly encouraged to also look at some comments that lead up to the answer.

An advertisement video from Piazza summarizes the features:

- <https://www.youtube.com/watch?v=2jLSiN8E18w>

Piazza Support with a lot of information is available at:

- <http://support.piazza.com>

A good document about piazza is available at

- https://piazza.com/pdfs/piazza_product_introduction.pdf

9.1 Access to Piazza from Canvas

Piazza is one of the recommended IU supported technologies within CANVAS. It replaces the CANVAS discussion groups with superior technology targeted to support large student classes while also focussing on student engagement.

To access piazza you can have the following situations provided in the next four subsections. Please read *ALL** of them **CAREFULLY**, decide which applies to you and follow the instructions. If you have improvements to this instructions, please let us know.

Situation: You have never logged into piazza

First, Click the Piazza link on the left navigation of your Canvas course.

The screenshot shows the Piazza interface within a Canvas course. On the left is a navigation sidebar with links like Home, Assignments, Discussions, Grades, People, Google Drive, Syllabus, **Piazza**, Quizzes, Conferences, Collaborations, Chat, Campus Course Policies, and Kaltura: Media Gallery. The main content area has a blue header with the Piazza logo and the text 'Join the discussion!'. Below this are three sections: 'Confirm Enrollment' (showing school, class, term, and role), 'My Account' (with fields for name, email, and password), and 'Welcome to Piazza!' (with a description of the platform and a link to 'See why Piazza works'). At the bottom of the 'My Account' section, there is a checkbox for 'I have read and I agree to Piazza's Terms of Use.' and a 'Continue' button.

Second, create password and accept terms.

The email address shown on this screen is your default IU email address. It is the address Canvas sends to all integrated tools like Piazza. You can not edit it, so do not try.

The password you create here is for accessing Piazza from a mobile device. You must use the default IU email address from this screen to access this account on another device, so make a note of it.

This is a close-up of the 'My Account' section from the previous screenshot. It shows the following fields and text:

- Your Name:** Carrie Hansel
- Your Email:** cahansel@iu.edu
- Text: You can access Piazza on the go with the Android and iOS apps, or directly from piazza.com. We need you to set a password, so you can log in.
- Choose a new Piazza password:** [password field]
- Confirm new password:** [password field]
- Checkbox: I have read and I agree to Piazza's [Terms of Use.](#)
- Button: Continue

Choose current degree program (only important if you want to opt into their recruiting program on the next screen; choose whatever you want here)

Third, associate your IU account

Forth, if all goes well you see the Success screen

Situation: You have logged into piazza and used your default IU e-mail

1. Click the Piazza link on the left navigation of your Canvas course.
2. You will be automatically enrolled in the course Piazza site and logged in.
3. Start using Piazza.

Situation: You have logged into piazza and you used another non IU e-mail

1. Click the Piazza link on the left navigation of your Canvas course.
2. Proceed as in #1 above. This will create your new Piazza account that is linked to your courses in Canvas. This is the account you should always use in your IU courses.
3. If you wish to merge other accounts that you own, please see [Add an email address or merge two accounts](#).

Situation: You have multiple accounts in piazza

1. If one of your multiple accounts corresponds with your default IU email address, you will be automatically enrolled in the course Piazza site and logged in.
2. If none of your accounts corresponds to your default IU email address, follow the instructions in #3 above.
3. If you wish to merge other accounts that you own, please see [Add an email address or merge two accounts](#).

I post the official response form the CANVAS team here: “When a student clicks the Piazza link

in your course navigation, they will be authenticated through to Piazza. If the student already has a Piazza account that matches their default Canvas email, they will simply be enrolled in the Piazza course. If the student doesn't have an account, Canvas sends the pertinent information (default email address primarily) to Piazza, Piazza creates the student's account and enrolls the student in the Piazza course. There is nothing you need to do.'

If you have any questions regarding accessing piazza, please send them to

'Ricci, Margaret P' <mricci@iu.edu>

9.2 Verify you are on Piazza via a post

Post on the **bio** folder a short introduction about yourself. One that you could include in a paper.

An example is provided at <https://laszewski.github.io/bio.html> with an image at https://laszewski.github.io/_images/gregor.jpg

Use the subject line *Biography: Firstname Lastname* and post it into the bio folder.

9.3 Making Piazza Work

In order for Piazza to work students and instructors need to participate

Students participate: Students must collaboratively work on an answer to a question. Students must not post irrelevant followups to a question. If you notice your comment was irrelevant, please delete it. Students must **search** prior to asking a new question if the question has already been asked. Duplicated questions can be merged.

Instructors guide: The instructor guides the students in order to obtain an answer to a question. In some cases the instructor may be the only one knowing the answer in which case he tries to provide it.

Not using e-mail: Instructors will and must not use e-mail to communicate with a student. All communication will be done via piazza. There, are only very view situations where e-mail is allowed, ask on piazza first if you should engage in e-mail conversations.

Not using CANVAS discussions: We will not engage in any CANVAS message exchange. Any communication is to be done on Piazza. It is in your responsibility to enroll in Piazza to make it work for you. Instructions are posted in this document. (Grade discussion will be done in CANVAS)

9.4 Towards good questions

Naturally when you ask a question you need to do it in a reasonable form and provide sufficient information so that the question can be answered. It is in the responsibility of the student to update the question to provide enough information.

Thus information may include: Firstname Lastname, HID, and URL to document in question

To give you an example of a **bad** question consider:

send from Xi Lee

Hi Professor:

I read a nice article about apples and potato's and updated my paper. Please give me feedback

Thank you

Kevin

Here the reasons why this can be improved:

1. As professors and instructors may review your document it is unnecessary to start with “Hi Professor:”, just leave it away. If you want a particular instructor use the name explicitly, such as “Gregor:”, e.g. multiple professors may be teaching your course.
2. You have not specified which article you read, you need to include the URL to the article so we can follow your argument.
3. You have not included the link to your document so we do not know what you are talking about. Remember there are many others students in the class
4. You are using a different name from the one that you are registered with. This can lead to confusion when we look up your name. We prefer that you use only one name that is associated with your e-mail.

The above question will simply be commented on (if at all):

“Missing information” or “?” indicating that information is missing.

It is in your responsibility to figure out which information is missing. YOU need to modify the original post and.

9.5 Guide on how to ask good questions

This guide is adapted from

- <http://www.techsupportalert.com/content/how-ask-question-when-you-want-technical-help.htm>

Ten steps to getting your question answered on piazza

1. Before you even go to ask a question, think through what your problem is. Write down how you are going to describe it. Think about it from the other side - what would you need to know if a student came to you and asked the question? Gather all the system information that seems to bear on the problem (see how at this link). Sometimes it even happens that by thinking through the problem, you come up with the answer yourself.
2. Verify that your question has not yet been answered with a search on the Web, Class Web page, or class piazza, this may require multiple searches.
3. In case it is a technical question, write down any error codes that appear on your screen. **Do not use screenshots** if the text is characters. This is because a reply may need to paste and copy from the original. Also screenshots are not searchable. We will not answer any questions that post screenshots if they are not necessary. It is far easier to copy and paste and use terminal type in the formatting. Also if the text is posted it is searchable. (Any unnecessary screenshot will receive a point deduction. Based on experience we have to do this as previous students in other classes ignored this policy).
4. Place your question or problem in a forum that is relevant to its subject. That may seem obvious but anyone who has experience with forums knows that a lot of questions show up in the wrong place. YOU will need to identify one or more a fitting piazza “folders” (folders sort the posts by topics).

5. Select a title that briefly and accurately describes your problem. A title like “Help!” or “Computer won’t work” will often get ignored. Almost any problem can be titled with a few key words that will raise interest in somebody who is familiar with the subject. A corollary to this is to avoid using all caps or a lot of exclamation points. Something like “HELP!!!” turns many people off.
6. In the post, briefly describe the problem in a paragraph. Leave out unnecessary details. Save everybody time by listing any solutions that you have tried but didn’t work. Avoid using screenshots if they are not needed. (I mention this again).
7. IN case of a technical issue describe relevant system details. For example, it is essential to designate your operating system and type of computer and any components that might be involved in your problem. List any error code that has been displayed. Be prepared to provide more details if asked.
8. Tell what you were doing when you encountered the problem. If it is a reproducible problem, list the steps or computer operations that cause the problem.
9. If applicable, List any recent software you have installed or hardware changes you have made. If you have updated any drivers recently, also list that.
10. Formulate your questions and answers in a courteous manner. Respect the answers from others. Somebody is giving you their time and expertise for free. You may want to come back to the forum and it pays to be friendly.
11. If a suggested solution works, be sure to return to piazza and report your success. It is the least you can do to return something for the help you have been given. It will make you welcome in the forum the next time you go there for help.

9.6 Piazza class Links

Using the following direct links can lead to you not getting proper access via Canvas. If you click on these links **before they create** the account via the link in your current Canvas course, you will create an account that is not matched up with Canvas.

To avoid issues make sure you integrate to piazza via Canvas first.

If you have questions bout this contact Margaret Ricci.

Classes hosted on Piazza

9.6.1 Current Classes

- E516 Spring 2018: <https://piazza.com/iu/spring2018/e516spring18/home>
- E616 and I524 Spring 2018: <https://piazza.com/iu/spring2018/e616spring18/home>

9.6.2 Previous Classes

- I523 Fall 2017: <https://piazza.com/iu/fall2017/i523/home>
- I524 Spring 2017: <https://piazza.com/class/ix39m27czn5uw>
- I523 Fall 2016: <https://piazza.com/class/irqfvh1ctrq2vt>

9.7 Piazza Curation

We are using Piazza in a curated fashion and we like that all students participate in this. This will allow Piazza to become a superior tool for all in the class. In general we only allow **exactly one folder** for a message. If a message is wrongly filed it will be corrected, either by students or TAs.

As part of this we are introducing a number of folders. Some of which must not be used by students. We list the following folders and their purpose:

Folder	Description
logistics	Any question and discussion related to the logistics of the course
lectures	Any question and discussion related to the lectures.
p1	Any question and discussion related to paper 1.
p2	Any question and discussion related to paper 2.
t1	Any question and discussion related to paper 1.
t2	Any question and discussion related to paper 2.
project	Any question and discussion related to iot projects.
term-paper	Any question and discussion related to the term project.
python	Any question and discussion related to python.
pi	Any question and discussion related to the Raspberry Pi 3. We are not using older Raspberry Pi's and therefore can not comment to them.
8266	Any question and discussion related to the esp8266.
bio	A homework folder in which you only publish your bio. The bio needs to be published as a <i>note</i> . This assignment also serves us to see if you are in piazza. Please do this assignment ASSAP. You need to post a formal bio. See the many great examples in the folder.
help	If you need help and none of the other folders fits, please use this folder. If information from here will result into new Web page content it will be added and marked into the folder <i>resolved</i> . See the <i>resolved</i> folder for more detail.
resolved	Sometimes we move some general help messages to the resolved folder in case the help message results into information that is posted on our class Web page. We than will add a link to where in the class Web page this question was answered. The TAs will aggressively try to put information into the Web page.
discussion	Any content that deserves its separate discussion and is not covered in the above folder.

In addition to these general folders we also have two folders which **MUST NOT BE USED BY ANY STUDENT TO POST CONTENT**. These folders serve to communicate your assignments and are used internally between Grgeor and the TA's.

assignments: This folder only lists the assignments. At any time in the class you can click on the assignment folder and list the assignments given to the class. Thus there is no confusion which assignments have been given. In case students have questions about assignments they should not use the *assignments* folder, but the *help* folder. TAs are instructed to correct wrongly filed messages in folders.

ta: Any question and discussion you have for the ta's. Typically you should however use the folder *help*. Gregor use most often the *ta* folder for internal coordination with the tas.

It may be necessary to create new folders for the class. Their meaning will be updated here once this occurs.

In case you decide to post privately and the information is useful for others also, the message will be published to the class.

9.8 Read the Originals, not just the e-mail

Piazza provides a convenient mechanism to update you through e-mail when an answer is changed or when someone posts.

However, this is just a reminder that something happened. In some cases I always recommend that instead of only reading the mail to use the <click here> feature in the mail to get not only the update, but to the actual post. Then you can get reminded about the information that is part of the post and potentially answers your question in full. It is not sufficient to participate in this class while only reading email, you should participate while visiting piazza and actively contribute to it.

9.9 Exercises

Exercise 9.1 Enroll in piazza ■

Exercise 9.2 Post a short formal bio in the bio folder and optionally include a professional portrait of yourself. Make sure you understand what a formal bio and portrait is. Research this in the internet. Look at IEEE papers for examples. ■

Exercise 9.3 How do you find out within Piazza which assignments have been posted? ■

Exercise 9.4 Please watch the Video about Piazza ■

9.10 Fall 2018**9.10.1 E534**

The following classes will be taught in Fall 2018:

- <http://registrar.indiana.edu/browser/soc4188/ENGR/ENGR-E534.shtml>

ENGR-E 534	BIG DATA APPLICATIONS (3 CR)									
*****	RSTR	ARR	ARR	ARR	Von Laszewski	G	30	30	0	
	Above class open to graduates only									
	Above class taught online									
	Discussion (DIS)									
14891	RSTR	09:30A-10:45A	F	I2 130	Von Laszewski	G	30	30	0	
	Above class meets with INFO-I 423 and I 523									

- <http://registrar.indiana.edu/browser/soc4188/INFO/INFO-I523.shtml>

9.10.2 I523

INFO-I 523	BIG DATA APPLS & ANALYTICS (3 CR)									
*****		ARR	ARR	ARR	Von Laszewski	G	10	10	0	
	Above class open to graduates only									
	Above class taught online									
	Discussion (DIS)									
11857	RSTR	09:30A-10:45A	F	I2 130	Von Laszewski	G	10	10	0	
	Above class meets with INFO-I 423 and ENGR-E 534									
INFO-I 523	BIG DATA APPLS & ANALYTICS (3 CR)									
11858	RSTR	ARR	ARR	ARR	Von Laszewski	G	90	90	0	
	Above class open to Data Science majors only									
	This is a 100% online class taught by IU Bloomington. No on-campus class meetings are required. A distance education fee may apply; check your campus bursar website for more information									
	Above class for students not in residence on the Bloomington campus									


9.10.3 I423

- <http://registrar.indiana.edu/browser/soc4188/INFO/INFO-I423.shtml>

INFO-I 423	BIG DATA APPLS & ANALYTICS (3 CR)									
*****	RSTR	ARR	ARR	ARR	Von Laszewski	G	10	10	0	
	Above class open to undergraduates only									
	Above class taught online									
	Discussion (DIS)									
12403	RSTR	09:30A-10:45A	F	I2 130	Von Laszewski	G	10	10	0	
	Above class meets with INFO-I 523 and ENGR-E 534									

10 Documenting Scientific Research

10	Documenting Scientific Research	145
10.1	Overview	
10.2	Plagiarism	
10.3	Acknowledgements	
10.4	Writing a Scientific Article or Conference Paper	
11	Introduction to \LaTeX	157
11.1	Installation	
11.2	Basic LaTeX Elements	
11.3	Advanced topics	
11.4	Editing	
11.5	The LaTeX Cycle	
11.6	Tips	
12	Managing Bibliographies	177
12.1	Integrating Bibliographies	
12.2	Entry types	
12.3	Integrating Bibtex entries into Other Systems	
12.4	Other Reference Managers	
13	Editors	195
13.1	Basic Emacs	
14	Other Formats	199
14.1	reStructuredText	
14.2	Markdown	
14.3	Communicating Research in Other Ways	



10. Documenting Scientific Research

10.1 Overview

Using a paper as part of your project planning is an important learning outcome. Instead of starting with a project we recommend that you start with a paper to direct your research.

This argument is made also by the following presentation.

How to write a paper by Simon Peyton Jones (57:39) 


We do recommend that you read the sections in this part carefully as they will introduce you to important tools that make writing a paper relatively simple while allowing professional paper format and bibliography management tools.

To get a first impression we have also prepared a number of videos that may help you. However, note that the format for papers used in these videos is different from the class and you must use the written documentation instead and use that format. Paper not using our format will be returned without review. I suggest you start right from the beginning.

Warning

The videos that show you the ACM paper template that we do not use

ShareLaTeX (8:49) 

jabref (14:41) 

Exercise 10.1 Watch the three lectures about How to write a paper, ShareLaTeX, and jabref. ■

10.2 Plagiarism

We start with the review of a most important topic.

10.2.1 Plagiarism Definition

In academic life it is important to understand and avoid plagiarism. The dictionary defines plagiarism as follows dictionary.com:

plagiarism “the practice of taking someone else’s work or ideas and passing them off as one’s own.”

10.2.2 Plagiarism Policies

Organizations and universities will have policies in place to address plagiarism. An example is provided for Indiana University [669]. We quote:

“Honesty requires that any ideas or materials taken from another source for either written or oral use must be fully acknowledged. Offering the work of someone else as one’s own is plagiarism. The language or ideas thus taken from another may range from isolated formulas, sentences, or paragraphs to entire articles copied from books, periodicals, speeches, or the writings of other students. The offering of materials assembled or collected by others in the form of projects or collections without acknowledgment also is considered plagiarism. Any student who fails to give credit for ideas or materials taken from another source is guilty of plagiarism.

(Faculty Council, May 2, 1961; University Faculty Council, March 11, 1975; Board of Trustees, July 11, 1975)”

Faculty members at Universities are also bound by policies that mandate reporting. At Indiana University the following policy applies (for a complete policy see the Web page):

“Should the faculty member detect signs of plagiarism or cheating, it is his or her most serious obligation to investigate these thoroughly, to take appropriate action with respect to the grades of students, and *in any event* to report the matter to the Dean for Student Services [or equivalent administrator]. The necessity to report every case of cheating, whether or not further action is desirable, arises particularly because of the possibility that this is not the student’s first offense, or that other offenses may follow it. Equity also demands that a uniform reporting practice be enforced; otherwise, some students will be penalized while others guilty of the same actions will go free.

(Faculty Council, May 2, 1961)”

Naturally if a student has any questions about understanding plagiarism the University can provide assistance. If a student is in doubt and asks for help this is not considered at that time plagiarism.

As you can see from the previous policies, the faculty do not have any choice but reporting real cases of plagiarism to the university administration. Thus you must not hold them personally responsible as this is part of the tasks they are required to do if they like it or not. Instead, it is **the responsibility of the authors of the document** to assure no plagiarism occurs. If you are a student of a class that writes a paper or project report this naturally also all applies to you. In addition, if you work in a team you need to assure the entire team addresses plagiarism appropriately.

In practice this means that the teachers of a course expect you know plagiarism and you need to be informed about it. This is typically done in other courses. However, as it is often overlooked by the

student we are pointing it out here so we can make sure you contribute to courses that require you to write papers and reports. This also means you can not claim you did not know what plagiarism is. You are required to know what it is, know how to detect it and know how to avoid it. The resources provided next will give you the necessary tools and background.

10.2.3 Plagiarism Resources

The [School of Education at Indiana University](https://www.indiana.edu/~istd/patterns.html) has a significant set of resources to get educated about plagiarism. These resources are intended to “preparing educators, advancing knowledge, and improving education” [669] <https://www.indiana.edu/~istd/patterns.html>

The content here is copied from the Web Page

- <https://www.indiana.edu/~istd/patterns.html>

As such we have not included quotes but refer to their Web page for the original source which may also include updates. Naturally we do not want to be accused of plagiarize in a chapter about plagiarism. Thus assume the content for the rest of this chapter are copied from that Web page. The resources in particular include:

- [IU Definition](#) of Plagiarism from Student Code of Conduct
- [Overview](#) How to give proper credit, steps.
- [Cases](#) of Plagiarism in the US, in the news, and elsewhere
- [Examples](#) Word for word, paraphrasing
- [Practice](#) with feedback on word-for-word and paraphrasing plagiarism
- [Test](#) 10 questions on recognizing plagiarism
- [Tutorial Site Map](#) Expanded table of contents
- [Resources](#) Websites, books, dictionary links, references for learning more about plagiarism

10.2.4 Tutorials

A number of tutorials are offered by Indiana University [Instructional Systems Technology Department](#) Web pages dealing with plagiarism. These include:

- [Plagiarsim Tutorial](#)
- [Understanding Plagiarism](#)

10.2.5 How to Recognize Plagiarism

We are listing fifteen patterns of plagiarism that are defined on the Web pages identified in Section 10.2.4 as part of the tutorials that we recommend you take:

Name	Plagiarism Type	Reason
Clueless Quote	word-for-word	no quotes, no citation, no reference
Crafty Cover-up	proper paraphrase but word-for-word	also present
Cunning Cover-up	paraphrasing	no citation, no reference
Deceptive Dupe	word-for-word	no quotes, no citation, but has reference
Delinked Dupe	word-for-word	no reference, even though quotes and citation
Devious Dupe	correct quote but word-for-word	also present
Dippy Dupe	word-for-word	quotes missing, even though full citation and reference
Disguised Dupe	looks like proper para, but actually word-for-word	no quotes, no locator
Double Trouble	word-for-word and paraphrasing	although has reference
Linkless Loser	word-for-word	citation and reference lacking, although has quotes and locator
Lost Locator	word-for-word	missing locator, although has quotes, citation, and reference
Placeless Paraphrase	paraphrasing	no reference, although citation present
Severed Cite	paraphrasing	reference but no citation
Shirking Cite	word-for-word	lacks locator and reference, although quotes and citation present
Triple D--Disguised Disconnected Dupe	word-for-word	looks like proper para, but no quotes, no reference, no locator

In addition they do specify three patterns of non-plagiarism:

Name	Type	Description
Correct Quote	non-plagiarizm	takes another's words verbatim and acknowledges with quotation marks, full in-text citation with locator, and reference
Proper Paraphrase	non-plagiarizm	summarizes another's words and acknowledges with in-text citation and reference
Parroted Paraphrase	non-plagiarizm	appears to be paraphrasing, and technically may not be plagiarism . . .

10.3 Acknowledgements

In many cases you not only want but must to include an acknowledgement section. In some cases you may be tempted to eliminate this section as you think you are out of space and the acknowledgement section may give you some additional space. This however is the wrong strategy and you should not do this. Instead you should shorten your paper elsewhere and leave enough space for acknowledgements.

In some cases where you get financial support from a university or a funding agency for a project such as from NIH or NSF specific information **must** be included. The best way is to verify with your coauthors. Additional acknowledgements may have to be added and you need to evaluate if for example significant help on the paper or the work that lead up to the paper warrants co-authorship.

An issue that we have seen often is for example when a professor has helped significantly on the paper but is not properly acknowledged. This can even lead to the professor asking you to remove him from the acknowledgement. A bad acknowledgement example is the following:

We like to thank Professor Zweistein for his help in compiling the \LaTeX paper.

We do not think that the professor will be happy with this acknowledgement as it sounds like the only thing that was provided was the help on \LaTeX that you should have done anyways without the help of the professor. Ask yourself, if he introduced you to the field, has helped you with preparing the text, has given you insights, has corrected things in your paper, made suggestions. So instead of the above maybe a more general term such as *helped with the paper* would be more appropriate. If not leaving it off is more appropriate. In some cases you may want to invite your professor to become a co-author. In some cases you may want to even include this handbook as a citation.

10.4 Writing a Scientific Article or Conference Paper

An important part of any scientific research is to document it. This is often done through scientific conferences or journal articles. Hence it is important to learn how to prepare and submit such papers. Most conferences accept typically the papers in PDF format but require the papers to be prepared on MSWord or in \LaTeX . While working with many students in the past we noticed however that those students using Word often spend unnecessarily countless hours on trying to make their papers beautiful while actually violating the template provided by the conference. Furthermore, we noticed that the same students had issues with bibliography management. Instead of Word helping the student it provided the illusion to be easier than \LaTeX but when adding up the time spent on the paper we found that \LaTeX actually saved time. This has been especially true with the advent of collaborative editing services such as `sharelatex` [569] and `overleaf` [482].

In this section we provide you with a professional template that is used based on the ACM standard that you can use to write papers. Naturally this will be extremely useful if the quality of your research is strong enough to be submitted to a conference. We structure this section as follows. Although we do not recommend that you use MSWord for your editing of a scientific paper, we have included a short section about it and outline some of its pitfalls that initially you may not think is problematic, but has proven to be an issue with students. Next we will focus on introducing you to \LaTeX and showcasing you the advantages and disadvantages. We will dedicate an entire section on bibliography management and teach you how to use `jabref` which clearly has advantages for us.

Having a uniform report format not only helps the students but allows instructors to integrate the comparison of paper length and effort as part of teaching a course. We have added an entire section to this chapter that discusses how we can manage a *Class Proceedings* from papers that are contributed by teams in the class.

10.4.1 Professional Paper Format

The report format we suggest here is based on the standard ACM proceedings format. It is of very high quality and can be adapted for your own activities. Moreover, it is possible to use most of

the text to adapt to other formats in case the conference you intend to submit your paper to has a different format. The ACM format is always a good start.

Important is that you do not need to change the template but you can change some parameters in case you are not submitting the paper to a conference but use it for class papers. Certainly you should not change the spacing or the layout and instead focus on writing content. As for bibliography management we recommend you use `jabref` which we will introduce in Section 12.1.

We recommend that you carefully study the requirements for the report format. We would not want that your paper gets rejected by a journal, conference or the class just because you try to modify the format or do not follow the established publication guidelines. The template we are providing is available from:

- <https://github.com/cloudmesh-community/hid-sample/tree/master/paper>

You will find in it a modified ACM proceedings templates that you must use.

10.4.2 Submission Requirements

Although the initial requirement for some conferences or journals is the document PDF, in many cases you must be prepared to provide the source when submitting to the conference. This includes the submission of the original images in an images folder. You may be asked to package the document into a folder with all of its sources and submit to the conference for professional publication.

10.4.3 Microsoft Word vs. \LaTeX

Microsoft word will provide you with the initial impression that you will save lots of time writing in it while you see the layout of the document. This will be initially true, but once you progress to the more challenging parts and later pages such as image management and bibliography management you will see some issues. These include that figure placement in Word needs to be done just right in order for images to be where they need. We have seen students spending hours with the placement of figures in a paper but when they did additional changes the images jumped around and were not at the place where the students expected them to be. So if you work with images, make sure you understand how to place them. Also always use relative caption counters so that if an image gets placed elsewhere the counter stays consistent. So never use just the number, but a reference to the figure when referring to it. Recently a new bibliography management system was added to Word. However, however it is not well documented and the references are placed in the system bibliography rather than a local managed bibliography. This may have severe consequences when working with many authors on a paper. The same is true when using Endnote. We have heard in many occasions that the combination of endnote and Word destroyed documents. You certainly do not want that to happen the day before your deadline. Also in classes we observed that those using LaTeX deliver better structured and written papers as the focus is on text and not beautiful layout.

For all these reasons we do not recommend that you use Word.

In LaTeX where we have an easier time with this as we can just ignore all of these issues due to relative good image placement and excellent support for academic reference management. Hence, it is in your best interest to use LaTeX. The information we provide here will make it easy for you to get started and write a paper in no time as it is just like filling out a form.

10.4.4 Working in a Team

Today research is done in potentially large research teams. This also include the writing of a document. There are multiple ways this is done these days and depends on the system you chose.

In MSWord you can use skydrive, while for LaTeX you can use sharelatex and overleaf. However, in many cases the use of github is possible as the same groups that develop the code are also familiar with github. Thus we provide you here also with the introduction on how to write a document in github while group members can contribute.

Here are the options:

LaTeX and git: This option will likely save you time as you can use jabref also for managing collaborative bibliographies and

sharelatex: an online tool to write latex documents

overleaf: an online tool to write latex documents

MS onedrive: It allows you to edit a word document in collaboration. We recommend that you use a local installed version of Word and do the editing with that, rather than using the online version. The online editor has some bugs. See also (untested): <http://www.paulkiddie.com/2009/07/jabref-exports-to-word-2007-xml/>, <http://usefulcodes.blogspot.com/2015/01/using-jabref-to-import-bib-to-microsoft.html>

Google Drive: google drive could be used to collaborate on text that is then pasted into document. However it is just a starting point as it does not support typically the format required by the publisher. Hence at one point you need to switch to one of the other systems.

10.4.5 Time Management

Obviously writing a paper takes time and you need to carefully make sure you devote enough time to it. The important part is that the paper should not be an after thought but should be the initial activity to conduct and execute your research. Remember that

1. It takes time to read the information
2. It takes time understand the information
3. It takes time to do the research

For deadlines the following will get you in trouble:

1. *There are still 10 weeks left till the deadline, so let me start in 4 week . . .* Procrastination is your worst enemy.
2. If you work in a team that has time management issues address them immediately
3. Do not underestimate the time it takes to prepare the final submission into the submission system. Prepare automated scripts that can deliver the package for submission in minutes rather than hours by hand.

10.4.6 Paper and Report Checklist

In this section we summarize a number of checks that you may perform to make sure your paper is properly formatted and in excellent shape. Naturally this list is just a partial list and if you find things we should add here, let us know.

One good way is to either copy the checklist into a file or print out just these pages and check with a pen if the particular issue occurs.

10.4.7 Content

- Is the paper formal paper and not an experience report?
- Do not include phrases such as ‘‘In week 1 we did this’’
- When writing the *proposal* do not use the word ‘‘proposal’’ write the document as if it would be the final paper. We see too many reports at the end forgetting to remove the word proposal in the final paper, so we can not tell if you did it or if it is still a proposal. As the final paper is not a proposal we reject such papers and you get a 1/3 grade reduction. To avoid this, just do not use the word proposal.
- When writing the abstract do not make it a proposal. Abstracts are no proposals. Avoid phrases such as We propose to do, We intend to show and so on. If the paper intends to show things you are still in the draft phase of the paper. However, if you say We show, that would be good. Let us just assume you intended to show something but did not achieve then you can say ‘‘We intended to show this but we it was not possible to verify. We have provided reasons for this in the paper’’. As you can see not only the intention is communicated, but the result. If you just focus on the intent that is just a proposal and is not a proper abstract.
- Add keywords to the paper, where the first two are your HID, and your class number.
- If your paper is an introduction or overview paper, please do not assume the reader to be an expert. Provide enough material for the paper to be useful for an introduction into the topic.
- If your paper limit is x number of pages but you want to hand in x plus 100 pages. If however your page limit is 2 pages and you hand in 4 or 6 pages that is no issue.

10.4.8 Submission

- Do not make changes to your paper during grading, when your repository should be frozen.
- Do not use filenames and directory names that have spaces in them only use [a-z0-9]*
- Make all file names lower case other than Makefile and README.yml
- You are required to run yamllint README.yml on all team members README.yml including your own. All of them must pass. Do this on the first day you start writing the paper. Only push and commit the files when they pass this test. If you do not have yamllint you can write one in python. Its 3 lines of code.
- Have you included the paper in the submission system (In our case git). This includes all images, bibliography files and other material that is needed to build the paper from scratch?
- Have you made sure your paper compiles with *make* and the provided Makefile before you committed?
- Are all images checked in?
- Did you submit the report.bib file?

10.4.9 Bibliography

- Are you managing your references in jabref and endnote (we need both)
- In the author field, authors are separated with an *and* and not a comma.
- The filename for the bibliography is report.bib.
- Bibtex labels must have any spaces, _ or & in it
- Fix citations in text that show as [?]. This means either your report.bib is not up-to-date or there is a spelling error in the label of the item you want to cite, either in report.bib or in report.tex
- Urls in citations are never placed in howpublished, instead we use url = { }. howpublished is just used for a text string such as Web Page, Blog, Repository and others like that. Do not use just the word Web, as it could be a Web Site or a Web Page. You need to be specific.

- Do not use the `\url={ }` in the text, instead use a citation.
- Are your references correct? References to a paper are no afterthought, they should be properly cited. Use `jabref` and make sure the citation type of the reference is correct and fill out as many fields as you can. Some journals and conferences have for example special requirements that go beyond the requirements of for example `jabref`. One example is that many conferences require you that when you cite papers from another conference to augment the conferences not only with the location where the conference took place, but also with the dates the conference took place. Unfortunately, this is information that is often only available through additional google queries and many reference entries you find in the internet do not have this information readily available.

10.4.10 Writing

- Have you spellchecked the paper?
- Have you grammar checked the paper?
- Use proper capitalization in the title, see: <https://capitalizemytitle.com/>
- Are you using *a* and *the* properly?
- Short form of verbs is for spoken language. Do not use them in scientific writing. Example: can't is incorrect, cannot is correct.
- Do not use phrases such as *shown in the Figure below*. Instead, use *as shown in Figure 3*, when referring to the 3rd figure, but use the `ref label` macros.
- Do not use the word *I* instead use *we* even if you are the sole author. In many cases you may want to avoid using the word *we* also.
- Do not use the phrase *In this paper/report we show* instead use *We show*. It is not important if this is a paper or a report and does not need to be mentioned.
- If you want to say *and* do not use *&* but use the word *and*.
- Use a space after `.`, `,`, `:`
- When using a section command, the section title is not written in all-caps as the \LaTeX template will do this automatically for you. Thus it is `\section{Introduction}` and NOT `\section{INTRODUCTION}`.

10.4.11 Citation Issues and Plagiarism

- It is your responsibility to make sure no plagiarism occurs.
- When stating claims you added the proper citations.
- Do avoid paraphrasing long quotations (whole sentences or longer) from other papers.
- Double check your paper if you have quote from other papers and included the citation.
- The `\cite{}` command must not be in the beginning of the sentence or paragraph, but in the end, before the period mark. Example: ... a library called Message Passing Interface (MPI) [7].
- Put a space between the citation mark and the previous word or better use `~`.
- There must not be any citation in the abstract or conclusion.
- Citations cannot be included in section headings they need to be included in the text.

10.4.12 Character Errors

The following errors are very often found and must be avoided.

- To emphasize a word, use *emphasize* and not “quote”. Quotes are reserved for quotes from other papers and must not be used to emphasize words or phrases. to put around a word that

you like to emphasize.

- Generally we do not use **bold fett** text. Instead use *em*.
- Erroneous use of quotation marks, i.e. use ‘ ‘ quotes ’ ’, but not the double quote that you find on your keyboard such as " ".
- When using the characters & # % _ put a backslash before them so that they show up correctly.
- Pasting and copying from the Web often results in non-ASCII characters to be used in your text, please remove them and replace accordingly. This is the case for quotes, dashes and all the other special characters.
- If you see a figure and not a figure in text you copied from a text that has the fi combined as a single character. It happens often with combinations of f such as fi fl ff

10.4.13 Structural Issues

- Does your paper include an Acknowledgement section.
- Is the acknowledgment including all the people appropriately that helped you in your activity.
- In case of a class and if you do a multi-author paper, you need to add an appendix called *Workbreak Down* describing who did what in the paper, after the bibliography
- Do you fulfill the minimum page length such as defined in the submission guideline. Remember that images, tables and figures do not count towards the page length.
- Do not artificially inflate your paper if you are below the page limit.
- In case you have an appendix it is included after the bibliography

10.4.14 Figures and Tables

- Images must be at least 300dpi if they are not in a scalable format such as PDF which you can generate from Powerpoint and other drawing programs.
- If you use Microsoft products, use ppt 4:3 ratio for drawing concent images. In case there is a powerpoint in the submission, the image must be exported as PDF.
- If you have OSX, you are allowed to use omnigraffle.
- Make sure you capitalize Figure 1, Table 2 when used in a sentence.
- Do use `\label{}` and `\ref{}` to automatically create figure numbers.
- Figure caption must be below the image.
- Table captions must be above the image.
- Do not include the titles of the figures in the figure itself but instead use the caption or that information.
- All images must be in native format, e.g. `.graffle`, `.pptx`, `.png`, `.jpg` in the images directory
- Do not submit eps images. Instead, convert them to PDF
- The image files must be in a single directory named *images*.
- Make the figures large enough so we can read the details. If needed make the figure over two columns
- Do not worry about the figure placement if they are at a different location than you think. Figures are allowed to float. To illustrate this case we force all images to be placed at the end of the paper, although you may have included it at a special location in the paper. This forces you to avoid the phrases as seen in the following image, but you need to use the ref and label features in LaTeX.
- In case you copied a figure from another paper you need to ask for copyright permission. In case of a class paper, you must include a reference to the original in the caption. In general we like to avoid this for the reports and like that you produce original pictures.

- Remove any figure that is not referred to explicitly in the text with a ref command. Agian just putting in the number will not be good enough. This allows you to place the figure in the final submission at a location without needing to fix the numbers.
- Do not use textwidth as a parameter for includegraphics, but instead use \columnwidth as demonstrated in our template.
- Figures should be reasonably sized and often you just need to add columnwidth e.g. `/includegraphics [width=1.0\columnwidth] {images/myimage.pdf}`
Do not play with the size, just leave it with 1.0.

If you observe something missing let us know.

10.4.15 Example Paper

An example report in PDF format is available:

- <https://github.com/cloudmesh-community/hid-sample/blob/master/paper-instructions.pdf>

10.4.16 Creating the PDF from LaTeX on your Computer

Latex can be easily installed on any computer as long as you have enough space. Furthermore if your machine can execute the make command we have provided in the standard report format a simple

[Makefile](#)

that allows you to do editing with immediate preview as documented in the LaTeX lesson.

10.4.17 Draft: Class Specific README.md

For the class we will manage all papers via github.com. You will be added to our github at

- <https://github.com/cloudmesh-community>

Previously we used

- <https://github.com/bigdata-i523>

and assigned an hid (homework index directory) directory with a unique hid number for you. In addition, once you decide for a project, you will also get a project id (pid) and a directory in which you place the projects. Projects must not be placed in hid directories as they are treated differently and a class proceedings is automatically created based on your submission.

As part of the hid directory, you will need to create a README.md file in it, that **must** follow a specific format. The good news is that we have developed an easy template that with common sense you can modify easily. The template is located at

- <https://raw.githubusercontent.com/bigdata-i523/sample-hid000/master/README.md>

As the format may have been updated over time it does not hurt to revisit it and compare with your README.md and make corrections. It is important that you follow the format and not eliminate the lines with the three quotes. The text in the quotes is actually yaml. Yaml is a data format the any data scientist must know. If you do not, you can look it up. However, if you follow our rules you should be good. If you find a rule missing for our purpose, let us know. We like to keep it simple and want you to fill out the *template* with your information.

Simple rules:

- replace the hid number with your hid number.
- naturally if you see sample- in the directory name you need to delete that as your directory name does not have sample- in it.
- do not ignore where the author is to be placed, it is in a list starting with a -
- there is always a space after a -
- do not introduce empty lines
- do not use TAB and make sure your editor does not bay accident automatically creates tabs. This is probably the most frequent error we see.
- do not use any : & _ in the attribute text including titles
- an object defined in the README.md must have on a single type field. For example in the project section. Make sure you select only one type and delete the other
- in case you have long paragraphs you can use the > after the abstract
- Once you understood how the README.md works, please delete the comment section.
- Add a chapter topic that your paper belongs to

10.4.18 Exercises

Exercise 10.2 Install latex and jabref on your system ■

Exercise 10.3 Check out the report example directory. Create a PDF and view it. Modify and recompile. ■

Exercise 10.4 Learn about the different bibliographic entry formats in bibtex ■

Exercise 10.5 What is an article in a magazine? Is it really an Article or a Misc? ■

Exercise 10.6 What is an InProceedings and how does it differ from Conference? ■

Exercise 10.7 What is a Misc? ■

Exercise 10.8 Why are spaces, underscores in directory names problematic and why should you avoid using them for your projects ■

Exercise 10.9 Write an objective report about the advantages and disadvantages of programs to write reports. ■

Exercise 10.10 Why is it advantageous that directories are lowercase have no underscore or space in the name? ■



11. Introduction to L^AT_EX

Mastering a text processing system is an essential part of a researcher's life. Not knowing how to use a text processing system can slow down the productivity of research drastically.

11.1 Installation

LaTeX is available on all modern computer systems. A very good installation for OSX is available at:

- <https://tug.org/mactex/>

However, if you have older versions on your systems you may have to first completely uninstall them.

11.1.1 Local Install

Installing L^AT_EX is trivial, and is documented on the internet very well. However, it requires sufficient space and time as it is a large environment. For a full installation, a system such as TeX Live requires about 5.5 GB. In addition to L^AT_EX we recommend that you install jabref and use it for bibliography management.

Thus you will have the most of them on your system.

- pdflatex: the latex program producing pdf
- bibtex: to create bibliographies
- jabref: GUI application to bibtex files (<http://www.jabref.org/>)

Make sure you check that these programs are there, for example with the Linux commands:

```
which pdflatex
which bibtex
```

which `jabref` (on OSX you may have an icon for it)

If these commands are missing, please install them. For the newest documentation on installation of \LaTeX we recommend you look up the installation for your specific OS.

Install on Ubuntu 16.04

The easiest way to install it on ubuntu is to use the terminal and type in (make sure you have enough space):

```
sudo apt-get install texlive-full
```

One of the best editors for \LaTeX is emacs as you can also do bibliography management with it and not just \LaTeX . However, other editors are available including:

- Kile, TeXworks, JLatexEditor, Gedit \LaTeX Plugin, TeXMaker

Please look up how to install them if you like to use them. TeXMaker is popular, However I find the combination of emacs and latexmk superior. TeXmaker is installed with:

```
sudo apt-get install texmaker
```

Other installations:

- kile is installed by default
- <https://www.tug.org/texworks/> (Works on ubuntu, Windows, OSX)

\LaTeX for OSX

- <https://www.latex-project.org/get/>

\LaTeX for Windows

- <https://www.latex-project.org/get/>
- <https://miktex.org/howto/install-miktex>

Exercise 11.1 Evaluate if you can install \LaTeX on Windows. We suggest you start with miktex. Find out how to install gitbash and make as we have some makefiles. We also want to figure out how to instal latexmk

11.1.2 Online Services

Sharelatex

Share \LaTeX is an online, collaborative \LaTeX editor that makes the creation, preview, and sharing of \LaTeX documents easy through a web-based interface. Those that like to use \LaTeX , but do not have it installed on their computers may want to look at the following video:

Video: <https://youtu.be/PfhSOjuQk8Y>

Video with cc: <https://www.youtube.com/watch?v=8IDCGTFXoBs>

Share \LaTeX not only allows you to edit online, but allows you to share your documents in a group of up to three. Licenses are available if you need more than three people in a team.

IU Licensed ShareLaTeX

At IU we has a license for the ShareLaTeX service available to School of Informatics and Computing and Engineering students, faculty, and staff only on the Bloomington campus.

You can create a free ShareLaTeX account but the free accounts have limitations. Adding your account to the IU license will give you access to advanced features, including unlimited sharing. It will also allow GitHub integration. This however only works with the commercial github.com and not the IU Enterprise GitHub at github.iu.edu. As we require in our courses github.com you will be able to use it.

Please note that this license is only available to School of Informatics and Computing students, faculty, and staff on the Bloomington campus. Students must be enrolled in one of the SoIC degree programs on the Bloomington campus to be eligible. Students in other degree programs (even those taking SoIC classes) are not eligible.

If you want to use this service, please do and be aware of the following:

1. Go to the ShareLaTeX site and register. Please note that you **must** use either an @indiana.edu or @iu.edu email address when you register. If you use any other email address, we will not be able to add you to our site license. You are also required to use your IU passphrase as your ShareLaTeX password. Once you have registered, send an email to soichelp@indiana.edu asking to have your ShareLatex account added to the IU license.
2. In your request, you must include the following: The IU email address you used when you registered (which must be in either the @indiana.edu or @iu.edu domain) A statement indicating that you understand that the ShareLaTeX service cannot be used for any sensitive data
3. Note that the ShareLaTeX service is **not** qualified for any sensitive data. This includes all data in the Critical, Restricted, and University-Internal categories as defined in the Data Classifications Page.

Overleaf

Overleaf.com is a collaborative L^AT_EX editor. In its free version it has a very limited disk space. However it comes with a Rich text mode that allows you to edit the document in a preview mode. The free templates provided do not include ACM template, put you are allowed to use the OSA template.

Features of overleaf are documented at:

- <https://www.overleaf.com/benefits>

Paperia

We do not know where this service is located. However it offers similar services as ShareLatex and Overleaf.

- <https://papeeria.com/>

Quicklatex

For very smalll L^AT_EX code and for formulas, various online services exist. On of these services is QuickLaTeX.com. Its main focus is to convert LaTeX to mathematical formulas.

- <http://quicklatex.com/>

11.2 Basic L^AT_EX Elements

Often researchers may be initially overwhelmed with all the features that L^AT_EX provides. However, it is much simpler than you initially believe. In our L^AT_EX sections we introduced you towards using an article template. As a template is provided you can just look at the elements in that article and modify or copy them while adapting the content. Thus, it is more like filling out a form. You do not have to learn much and you can learn as you go. We are providing in this chapter some basic L^AT_EX elements that will help you getting started quickly while serving you as a reminder what how to do certain things in L^AT_EX.

11.2.1 Characters

LaTeX is a command language and as such uses some special characters as part of the language. Thus if you want to use these characters either in your text or bibliography you need to be especially careful about. These characters include % \$ # _

Other than in hyperref links and urls you need to put a backslash in front of them. For example to print a % in the text you need to use:

```
\%
```

Furthermore the character " is not at all used as discussed in the next section.

11.2.2 Highlighting Text

Quotes are not written with the " character, but are embedded in two left single quotes and two right single quotes:

quote	
<code>‘‘This is a quote’’</code>	‘‘This is a quote’’

In many papers we see that the quote is misused while putting quotes around a word. However quotes are often just used to quote a text from another paper. Instead of using quotes authors may actually emphasize a word. LaTeX has a special command for that using:

emphasize	
<code>\textit{this is emphasized}</code>	<i>this is emphasized</i>

To write a text as bold (which should also be avoided as bold is typically used in section headers), you can use:

bold fett	
<code>{\bf this is bold fett}</code>	this is bold fett

11.2.3 Sections

LaTeX provides a convenient mechanism to structure a paper with sections and subsections. This is achieved with the following commands:


```
\section{This is a Section}
\subsection{This is a Subsection}
\subsubsection{This is a Subsubsection}
```

Once you use one of these commands the next paragraph will start below the section command.

In addition you have the command:

```
\paragraph{This is a paragraph.}
```

The line is behind the paragraph heading

The command is special as it does not introduce a new line between the Heading and the next line even if you include empty lines.

11.2.4 Empty Lines

Multiple empty lines will be reduced to a single empty line.

11.2.5 Itemize

Itemized lists can be written as follows:

itemize	
<pre>1 \begin{itemize} 2 \item First item 3 \item Second item 4 \end{itemize}</pre>	<ul style="list-style-type: none">• First item• Second item

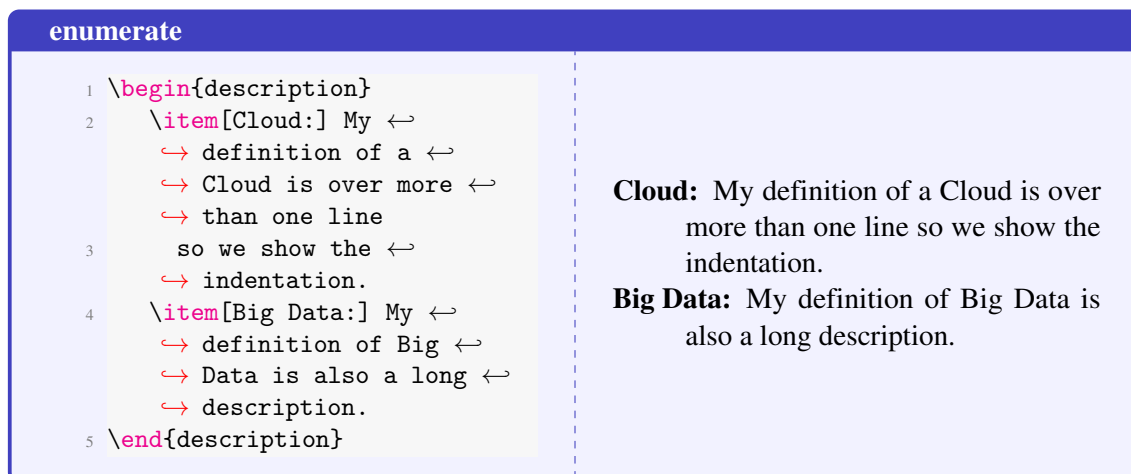
11.2.6 Enumerate

Enumerations can be written as:

enumerate	
<pre>1 \begin{enumerate} 2 \item First item 3 \item Second item 4 \end{enumerate}</pre>	<ol style="list-style-type: none">1. First item2. Second item

11.2.7 Descriptions

Description lists can be written as:



11.2.8 Images

Figures are extremely easy to handle by including them from source. We never worry about the placement as LaTeX does typically a very good job of doing this.:

In Figure~\ref{F:flow} we show a black and white graph about \ldots

```

\begin{figure}[htb]
  \includegraphics[width=1.0\columnwidth]{images/gregor.png}
  \caption{A demonstration in the scalability of PDF images.}\label{F:flow}
\end{figure}

```

Which results in the following:

In Figure 11.1 we show a black and white graph about ...

Note that las17graph must be a label of a valid bibtex entry. This is needed if you have copied the image from elsewhere to avoid plagiarism. However, if you came up with the graph yourself than you do not need a citation.

We recommend that you place in your paper drafts all images at the which can be done with the endfloat package

This can be enabled if you include the following lines before begin document command:

```

\usepackage{endfloat}
\renewcommand{\efloatseparator}{\mbox{}}

\begin{document}

```

11.2.9 Tables

Latex tables are very similar to csv table. Thus you could with appropriate manipulation create tables from csv tables and use tools such as spreadsheet editors to manage your table. There are even packages that allow you to import the contents of csv tables directly into L^AT_EX.

In many cases you simply can start with an online table generator such as

- <https://www.tablesgenerator.com/>

For some tables you may want to rescale the width with

```

\resizebox{\textwidth}{!}{%
... PUT YOUR TABLE HERE ....
}

```

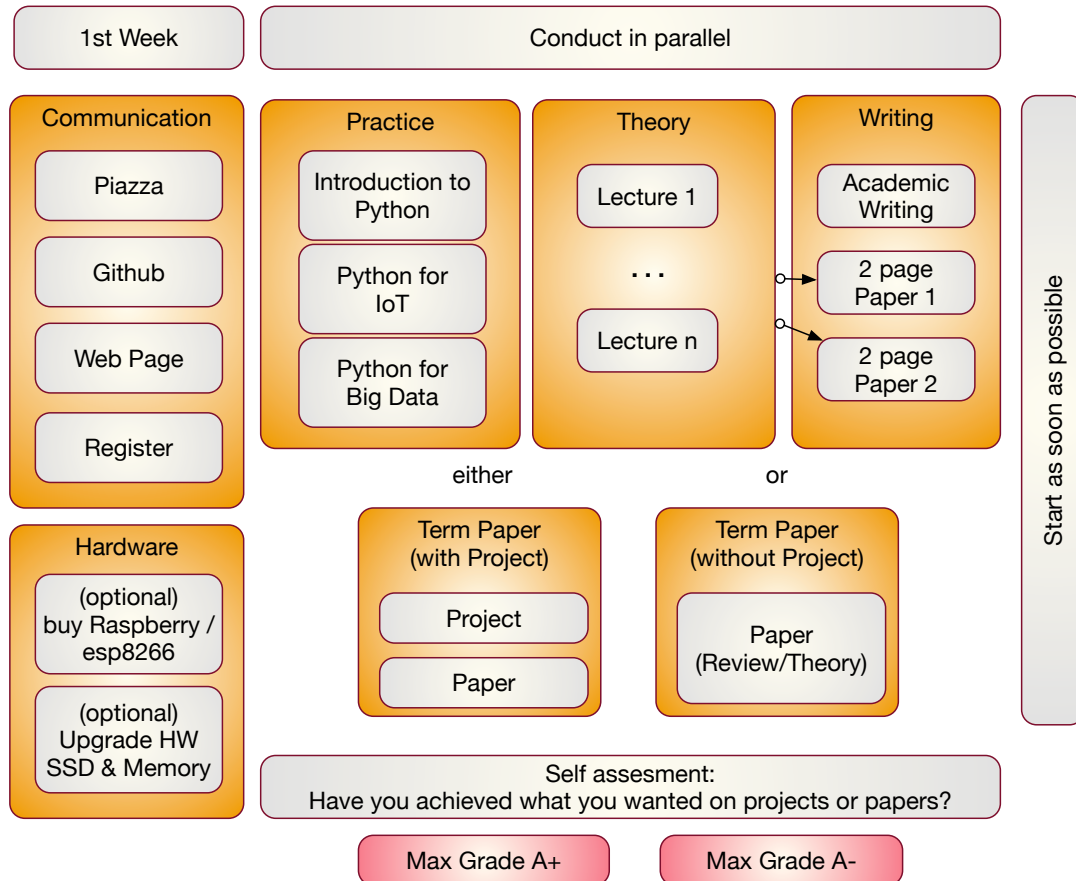



Figure 11.1: A demonstration in the scalability of PDF images.

}

This can even be autogenerated if you switch on in the tables generator the option `scale table to textwidth`. In other cases you may want to rotate the table which you can easily google for. In all cases use as figures, tables need to be in the `table float` environment. In contrast to figures captions are on the top. All tables must be referred to by `ref`. For more information in directly including csv tables see

- <http://mirror.utexas.edu/ctan/macros/latex/contrib/csvsimple/csvsimple.pdf>

However the format is very easy and can in most cases directly be included in latex as shown in Table 11.1.

```
\begin{table}[htb]
\caption{Table with elements}\label{T:elements}
\bigskip
\begin{center}
\begin{tabular}{c c c }
column1 & column2 & column3 \\
\toprule
element1 & element2 & element3 \\
\end{tabular}
\end{center}
\end{table}
```

```

    element4 & element5 & element6 \\
    element7 & element8 & element9 \\
\bottomrule
\end{tabular}
\end{center}
\end{table}

```

Table 11.1: Table with elements

column1	column2	column3
element1	element2	element3
element4	element5	element6
element7	element8	element9

11.2.10 Labels

As we saw already for figures and tables it is recommended to use the label and ref commands to refer to figure or table numbers. This applies also to sections. Thus I can place a label after a section:

```
\section{Introduction}\label{S:introduction}
```

and write elsewhere in the paper:

```
As we showcased in Section~\ref{S:introduction}
```

Furthermore to conveniently distinguish sections tables and figures, we use the prefix S T F followed by a colon for the label. This helps organizing your paper in case you have many labels.

11.2.11 Mathematics

One of the strength of LaTeX thi the ability to write easily sophisticated mathematical expressions on paper with high quality. A good online resource is provided by the following online resource from which we have copied some examples:

- <https://en.wikibooks.org/wiki/LaTeX/Mathematics>

To activate them use

```
\usepackage{amsmath}
```

at the beginning of the document after the document class

Exponents are using the ^ character:

exponents

<pre>1 \mathbf{\\$(a+b)^2 = a^2 + 2ab + \leftrightarrow} \mathbf{\red{<->} b^{c+2}\\$}</pre>	$(a + b)^2 = a^2 + 2ab + b^{c+2}$
---	-----------------------------------

Greek letters are referred to by their name proceeded by the slash:

greek

```
1 $ \alpha \beta \gamma \leftrightarrow
   \rightarrow \Gamma \pi \Pi \phi $
```

$$\alpha\beta\gamma\Gamma\pi\Pi\phi$$

Limits can be written as follows:

limits

```
1 $ \lim_{x \to \infty} \leftrightarrow
   \rightarrow \exp(-x) = 0 $
```

$$\lim_{x \rightarrow \infty} \exp(-x) = 0$$

Fractions are indicated by the frac command, and binomials by binom:

fraction

```
1 $ \frac{n!}{k!(n-k)!} = \leftrightarrow
   \rightarrow \binom{n}{k} $
```

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Matrices can be created as follows:

matrix

```
1 $ A_{m,n} =
2 \begin{pmatrix}
3 a_{1,1} & a_{1,2} & \dots & a_{1,n} \\
4 a_{2,1} & a_{2,2} & \dots & a_{2,n} \\
5 \vdots & \vdots & \ddots & \vdots \\
6 a_{m,1} & a_{m,2} & \dots & a_{m,n} \\
7 \end{pmatrix} $
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

11.3 Advanced topics

11.3.1 ACM and IEEE Proceedings Format

- <http://www.acm.org/publications/proceedings-template>
- https://www.ieee.org/conferences_events/conferences/publishing/templates.html

11.3.2 Generating and Managing Images

To produce high quality images the programs PowerPoint and omnigraffle on OSX are recommended. When using powerpoint please keep the image ratio to 4×3 as they produce nice size graphics which you also can use in your presentations. When using other ratios they may not fit in presentations and thus you may increase unnecessarily your work. We do not recommend vizio as it is not universally available and produces images that in case you have to present them in a slide presentation does not easily reformat if you do not use 4×3 aspect ratio.

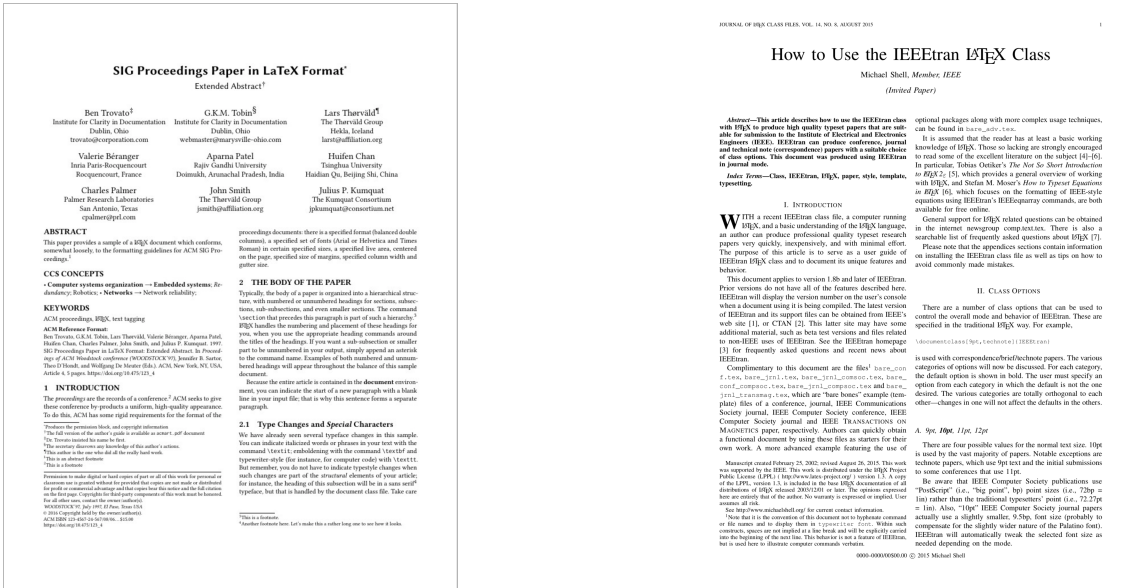


Figure 11.2: The look of the ACM and IEEE format templates

Naturally, graphics should be provided in SVG or PDF format so they can scale well when we look at the final PDF. Including PNG, gif, or jpeg files often do not result in the necessary resolution or the files become real big. For this reason we for example can also not recommend tools such as tablaeu as they do not provide proper exports to high quality publication formats. For interactive display such tool may be good, but for publications it produces inferior formatted images.

We recommend that all images be stored into a folder called images in the same directory where your L^AT_EX main document resides.

11.3.3 Colored Boxes

The package `tcolorbox` provides sophisticated support to include color boxes. Together with the new environment we can create nice add ons to for example include notes.

- <http://osl.ugr.es/CTAN/macros/latex/contrib/tcolorbox/tcolorbox.pdf>

We have provided in this document notes as follows

<p>Note</p> <pre>1 \begin{NOTE} 2 This is a note 3 \end{NOTE}</pre>	<p>Note</p> <p>This is a note</p>
--	--

<p>Warning</p> <pre>1 \begin{WARNING} 2 This is a note 3 \end{WARNING}</pre>	<p>Warning</p> <p>This is a note</p>
---	---

11.3.4 Slides

Slides are best produced with the seminar package:

```
\documentclass{seminar}

\begin{slide}

    Hello World on slide 1

\end{slide}

The text between slides is ignored

\begin{slide}

    Hello World on slide 2

\end{slide}
```

However, in case you need to have a slide presentation we recommend you use ppt. Just paste and copy content from your PDF or your LaTeX source file into the ppt.

11.3.5 LaTeX vs. X

We will refrain from providing a detailed analysis on why we use LaTeX in many cases versus other technologies. In general, we find that LaTeX:

- is incredibly stable
- produces high-quality output
- is platform independent
- has lots of templates
- has been around for many years so it works well
- removes you from the pain of figure placements
- focusses you on content rather than the appearance of the paper
- integrates well with code repositories such as git to write collaborative papers.
- has superior bibliography integration
- has a rich set of tools that make using LaTeX easier
- authors do not play with layouts much so papers in a format are uniform

In case you need a graphical view to edit LaTeX or LaTeX exportable files you also find AucTeX and Lyx.

Word

Word is arguably available to many, but if you work on Linux you may be out of luck. Also Word often focusses not on structure of the text but on its appearance. Many students abuse Word and the documents in Word become a pain to edit with multiple users. Recently Microsoft has offered online services to collaborate on writing documents in groups which work well. Integration with bibliography managers such as endnote or Mendeley is possible.

However, we ran into issues whenever we use word:

- Word tends sometimes to crash for unknown reasons and we lost a lot of work
- Word has some issues with the bibliography managers and tends to crash sometimes for unknown reasons.

- Word is slow with integration to large bibliographies.
- Figure placement in Word in some formats is a disaster and you will spend many hours to correct things just to find out that if you make small changes you have to spend additional many hours to get used to the new placement. We have not yet experienced a word version where we have not lost images. Maybe that has changed, so let us know

However, we highly recommend the collaborative editing features of Word that work on a paragraph and not letter level. Thus saving is essential so you do not block other people from editing the paragraph.

Google Docs

Unfortunately, many useful features got lost in the new google docs. However, it is great to collaborate quickly online, share thoughts and even write your latex documents together if you like (just copy your work in a file offline and use latex to compile it).

The biggest issue we have with Google Docs is that it does not allow the support of 2 column formats, that the bibliography integration is non-existent and that paste and copy from web pages and images encourages unintended plagiarism when collecting information without annotations. LaTeX and Word are prone to this too, but we found from experience that it tends to happen more with Google docs users.

A Place for Each

When looking at the tools we find a place for each:

Google docs: Short meeting notes, small documents, quick online collaborations to develop documents collaboratively at the same time.

Word: Available to many, supports 2 column format, supports paragraph based collaborative editing, Integrates with bibliography managers.

LaTeX: Reduces failures, great offline editing, superior bibliography management, superior image placement, runs everywhere. Great collaborative editing with sharelatex, allows easy generation of proceedings written by hundreds of people with shared index.

The best choice for your class: LaTeX

11.4 Editing

11.4.1 Emacs

The text editor emacs provides a great basis for editing TeX and LaTeX documents. Both modes are supported. In addition there exists a color highlight module enabling the color display of LaTeX and TeX commands. On OSX aquamacs and carbon emacs have build in support for LaTeX. Spell checking is done with flyspell in emacs.

Aquamacs

Aquamacs is an editor based on GNU Emacs that runs on OSX and integrates with the OSX desktop. This is for many the preferred editor on OSX for \LaTeX .

<http://aquamacs.org>

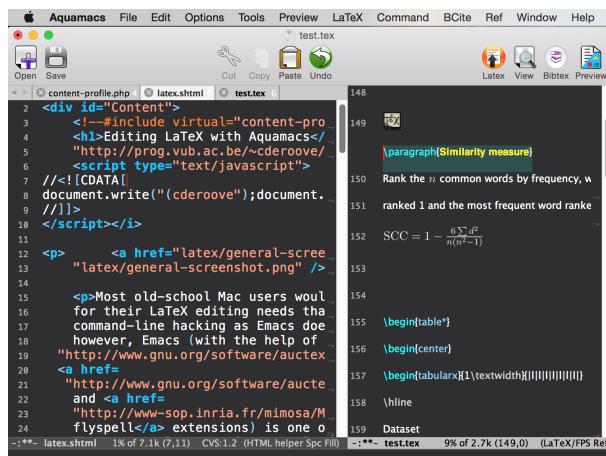


Figure 11.3: Aquamacs

11.4.2 Vi/Vim

Another popular editor is vi or vim. It is less feature rich but many programmers are using it. As it can edit ASCII text you can edit LaTeX. With the LaTeX add-ons to vim, vim becomes similar powerful while offering help and syntax highlighting for LaTeX as emacs does. (The authors still prefer emacs)

11.4.3 TeXshop

Other editors such as TeXshop are available which provide a more integrated experience. However, we find them at times to stringent and prefer editors such as emacs.

11.4.4 LyX

We have made very good experiences with LyX. You must assure that the team you work with uses it consistently and that you all use the same version.

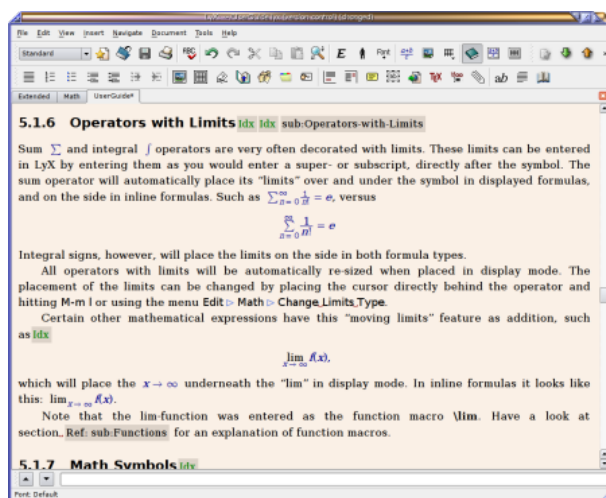


Figure 11.4: LyX

Using the ACM templates is documented here:

- <https://wiki.lyx.org/Examples/AcmSiggraph>

On OSX it is important that you have a new version of LaTeX and Lyx installed. As it takes up quite some space, you may want to delete older versions. The new version of LyX comes with the acmsigplan template included. However on OSX and other platforms the .cls file is not included by default. However the above link clearly documents how to fix this.

11.4.5 WYSIWYG locally

We have found that editors such as Lyx and Auctex provide very good WYSIWYG alike features. However, we found an even easier way while using skim, a pdf previewer, in conjunction with emacs and latexmk. This can be achieved while using the following command assuming your latex file is called 'report.tex':

```
latexmk -pvc -view=pdf report
```

This command will update your pdf previewer (make sure to use skim) whenever you edit the file report.tex and save it. It will maintain via skim the current position, thus you have a real great way of editing in one window, while seeing the results in the other.

Skim can be found at: <http://skim-app.sourceforge.net/>

11.4.6 Markdown and L^AT_EX

It may come as a surprise to many that one can actually write simple LaTeX documents also in markdown Syntax or mix section written in markdown while others are written in LaTeX. To do so all you have to do is place the markdown text in a separate file. Let us call the file content.md which has the following lines included in it:

```
# Section

* item a
* item b
```

Obviously, we would have to convert this to LaTeX. Luckily there is a very useful program called *pandoc* that does this for you. You could make the translation in the shell, but you could also make the translation locally on your computer while allowing L^AT_EX to start up external programs. This is achieved with the *write18* command and allowing LaTeX explicitly to call external programs. Please inspect the following latex file that includes a template on how to do this. We assume the file is called markdown.tex for our example.

```
\documentclass{article}

\include{graphicx}
\newcommand{}{}

\begin{document}
\immediate\write18{pandoc content.md -o content.tex}

\input{content}

\end{document}
```

Now to generate the PDF we simply have to call the following command that include the *-shell-escape* flag to allow the execution of write18 embedded commands:

```
pdflatex -shell-escape markdown-test
```


The output will be *markdown.pdf* with the content from the markdown file translated. Doing this naturally allows you to write large portions in markdown and automatically include them in your LaTeX document. Hence, you can use editors such as Maccdown to initially work in semi WYSIWYG mode and do fairly straight forward edition. Naturally the same can be done in RST. Naturally the most elementary features are supported. For more sophisticated features, please use LaTeX directly.

11.4.7 Including RST into LaTeX

content.rst:

```
Section
-----

* item a
* item b
```

sample.tex:

```
\documentclass{article}

\include{graphicx}
\newcommand{\tightlist}{}

\begin{document}
\immediate\write18{pandoc content.rst -o content.tex}

\input{content}

\end{document}
```

11.4.8 pyCharm

TODO: comment on how we can use pycharm for editing and what the limitations are.

11.4.9 MSWord

it is possible to use Word.

be careful with

11.5 The LaTeX Cycle

To create a PDF file from latex yo need to generate it following a simple development and improvement cycle.

First, Create/edit ASCII source file with `file.tex` file:

```
emacs file.tex
```

Create/edit bibliography file:

```
jabref refs.bib
```

Create the PDF:

```
pdflatex file
bibtex file
```

```
pdflatex file
pdflatex file
```

View the PDF:

```
open file
```

It not only showcases you an example file in ACM 2 column format, but also integrates with a bibliography. Furthermore, it provides a sample Makefile that you can use to generate view and recompile, or even autogenerate. A compilation would look like:

```
make
make view
```

If however you want to do things on change in the tex file you can do this automatically simply with:

```
make watch
```

for make watch its best to use skim as pdf previewer

11.6 Tips

Including figures over two columns:

- <http://tex.stackexchange.com/questions/30985/displaying-a-wide-figure-in-a-two-column-document>
- positioning figures with `textwidth` and `columnwidth` https://www.sharelatex.com/learn/Positioning_images_and_tables
- An organization as the author. Assume the author is National Institute of Health and want to have the author show up, please do:


```
key= {National Institute of Health},
author= {{National Institute of Health}},
```

 Please note the `{{ }}`
- words containing ‘fi’ or ‘ffi’ showing blank places like below after recompiling it: find as nd efficiency as e ciency
 You copied from word or PDF ff which is actually not an ff, but a condensed character, change it to ff and ffi, you may find other such examples such as any non ASCII character. A degree is for example another common issue in data science.
- do not use `|` & and other latex characters in bibtex references, instead use `,` and the word and
- If you need to use `_` it is `_` but if you use urls leave them as is
- We do recommend that you use sharelatex and jabref for writing papers. This is the easiest solution and beats in many cases MSWord as you can focus on writing and not on formatting.

11.6.1 Colorful Output

Instead of using `pdflatex`, you can also install `pydfplatex` that provides a convenient wrapper and colorizes the output while eliminating a lot of warnings that you may initially not want to deal with. To install it please use:

```
pip install blessings
pip install -e "git+https://github.com/olivierverdier/pydfplatex#egg=pydfplatex"
```

You can see the manual page with

```
pydfplatex --help
```

```
usage: usage: pydflatex [options] texfile1
```

Compile a tex file with pdflatex and make the auxiliary files invisible. Note that the '.tex' extension may be omitted

positional arguments:

```
  tex path          path to tex file
```

optional arguments:

```
-h, --help          show this help message and exit
-o, --open          view the pdf file(s) in a pdf viewer.
-k, --continue      continue on error
-w, --with-warning  do not suppress common warnings
-v, --verbose       Verbose output for debugging
-p, --plain         No coloured output
-x, --xetex        Use XeLaTeX engine
-l, --log-parsing  Only parse log
-t, --typesetting  Only typeset
```

11.6.2 latex2html

\LaTeX can be exported to html with the the tool latex2html. It is available for Linux and OSX. More information can be found at

- <http://mirrors.ctan.org/support/latex2html/manual.pdf>
- www.latex2html.org

Warning

At this time the inastallation via brew install of latex2html is broken. Instead one needs to conduct the insalation from source.

```
brew install latex2html
```

The instalation from source can be conducted as follows

```
brew install libpng
wget https://github.com/latex2html/latex2html/archive/master.zip
unzip master.zip
cd latex2html-master/
configure
make
make check
sudo make install
```

11.6.3 lacheck

Lacheck allows to finding common mistakes in \LaTeX documents. We recommend that you use it to check your latex files.

You can invoke it as follows

```
lacheck filename.tex
```

More information can be found at

- <https://ctan.org/tex-archive/support/lacheck>

11.6.4 chktex

LaTeX is a powerful program to develop professional documents. In some cases it is useful to check some semantics on the document. For this reason you can use

```
1 chktex filename.tex
```

It will execute a number of filters and produce a check on them. You can configure it to produce warnings and errors based on the error type.

For a manual page use

```
1 chktex --help
```

When using it some errors and warnings can be ignored, while others should be considered. Supported features mentioned on the Web page include: “

- Commands terminated with space. Ignores $\backslash tt$, etc.
- Space in front of references instead of \sim .
- Forgetting to group parenthesis characters when sub-/superscripting.
- Italic correction ($\backslash /$) mistakes (double, missing, unnecessary).
- Parenthesis and environment matching.
- Ellipsis detection; also checks whether to use $\backslash dots$, $\backslash cdots$ or $\backslash ldots$.
- Enforcement of normal space after abbreviation. Detects most abbreviations automatically.
- Enforcement of end-of-sentence space when the last sentence ended with capital letter.
- Math-mode on/off detection.
- Quote checking, both wrong types (") and wrong direction.
- Recommends splitting three quotes in a row.
- Searching for user patterns.
- Displays comments.
- Space in front of $\backslash label$ and similar commands.
- Use of x instead of $\$ \backslash times \$$ between numbers.
- Multiple spaces in input which will be rendered as one space (or multiple spaces, where that is undesirable).
- Warns about text which may be ignored.
- Mathematical operators typeset as variables.
- No space in front of/after parenthesis.
- Demands a consistent quote style.
- Punctuation inside inner math mode/outside display math mode.
- Use of TeX primitives where LaTeX equivalents are available.
- Space in front of footnotes.
- Bogus characters following commands.

“

Based on students reports that were sent to us, we found the following command returns the most significant issues:

```
1 chktex -n 13 -n 8 -n 29 filename.tex
```

For more details see

- <http://baruch.ev-en.org/proj/chktex/>

11.6.5 Chacking For Non-ASCII Charaters

To create the most portable \LaTeX we recommend that you avoid non-ASCII characters and replace them with the appropriate \LaTeX symbol. To find the characters a program such as this could be helpful. Important is that you not only check the tex file, but also the bib file.

```
1 perl -ane "{ if(m/[[:^ascii:]]/) { print } }" filename.tex
2 perl -ane "{ if(m/[[:^ascii:]]/) { print } }" filename.bib
```

11.6.6 References

Latex Sheet: <https://wch.github.io/latexsheet/latexsheet.pdf>

Latex Short: <http://tug.ctan.org/info/lshort/english/lshort.pdf>

Wikibook: <https://en.wikibooks.org/wiki/LaTeX>

Wikibook (PDF) : <https://upload.wikimedia.org/wikipedia/commons/2/2d/LaTeX.pdf>

Links to books: <https://latexforhumans.wordpress.com/2008/10/11/the-best-guides-to-latex/>

Links to books: <https://www.latex-project.org/help/books/>

LaTeX2e: The [LaTeX Reference Manual](#) provides a good introduction to Latex.

- LaTeX Users and Reference Guide, by Leslie Lamport https://www.amazon.com/LaTeX-Document-Preparation-System-2nd/dp/0201529831/ref=sr_1_2?s=books&ie=UTF8&qid=1507114870&sr=1-2&keywords=lamport
- LaTeX an Introduction, by Helmut Kopka https://www.amazon.com/Guide-Latex-4th-Helmut-Kopka/dp/0321173856/ref=pd_lpo_sbs_14_t_0?encoding=UTF8&psc=1&refRID=2BB4APDFEX34A4JM65ZB
- The LaTeX Companion, by Frank Mittelbach <https://www.amazon.com/LaTeX-Companion-Techniques-Computer-Typesetting/dp/0201362996>



12. Managing Bibliographies

12.1 Integrating Bibliographies

Bibliography management in \LaTeX is one of the many features that motivate many researchers to make \LaTeX the tool of choice to write academic papers. There are numerous bibliography styles available that allow easy adaptation to different journal and proceedings style. As such, it includes styles for bibliographies formatted in ACM and IEEE style.

12.1.1 biber

TODO: Describe use of biber instead of bibtex

12.1.2 bibtex

This section assumes we use the standard bibtex program to integrate citations into a \LaTeX paper. An example to use the IEEE style for a paper includes to set the style to IEEEtran before you add the reference:

```
\bibliographystyle{IEEEtran}
\bibliography{references.bib}
```

To properly create PDF papers which using bibtex, we have to make sure that all indexes, citations, references and labels are updated. This is done with the help of the following commands assuming your file is called `file.tex`

```
pdflatex file
bibtex file
pdflatex file
pdflatex file
```

Indiana University

At IU you are required to use our template, which includes a Makefile and either calls the above three commands or used latexmk

The reason for the multiple execution of the \LaTeX program is to update all cross-references correctly. In case you are not interested in updating the library every time in the writing progress just postpone it till the end. Missing citations are viewed as [?].

Two programs stand out when managing bibliographies: emacs and jabref:

- <http://www.jabref.org/>

Other programs such as Mendeley, Zotero, and even endnote integrate with bibtex. However their support is limited, so we recommend that you just use jabref as it is free and runs on all platforms.


12.1.3 jabref

TODO: Tyler: include cite for jabref, bibtex, biber at appropriate location in this section. Add them to the bib file.

Practical experience with many generations of students shows that *jabref* is a very simple to use bibliography manager for \LaTeX and other systems. It can create a multitude of bibliography file formats and allows upload in other online bibliography managers.

For information on how to install and use jabref please go to <http://www.jabref.org/>. There you will find the appropriate download link.

We provide also a convenient video on how to use jabref and show several methods on how to add entries easily to your bibliography database.

jabref (1:41) 

12.2 Entry types

In this section we will explain how to find and properly generate bibliographic entries. We are using bibtex for this as it is easy to use and generates reasonable entries that can be included in papers. What we like to achieve in this section is not to just show you a final entry, but to document the process on how that entry was derived. This will allow you to replicate or learn from the process to apply to your own entries. As part of this we copy and paste information found via Web searches.

We will address a number of important reference types which includes:

- wikipedia entries
- github (see Section 12.2.1)
- books
- articles in a scientific journal (see Section 12.2.3)
- articles in a conference (see Section 12.2.4)
- articles in magazines (non scientific)
- blogs

12.2.1 Source Code References

Often, we need to cite a source code from a publicly hosted repository. Such repositories are frequently used and include, for example github, bitbucket, sourceforge, or your Universities code repository as long as it is publicly reachable. As changes can occur on these repositories, it is important that the date of access is listed in the entry or even the release version of the source code.

Let us without bias chose a random source dode entry that has been contributed by a student as follows:

```
@Misc{gonzalez_2015,
  Title = {Buildstep},
  Author = {Gonzalez, Jose and Lindsay, Jeff},
  HowPublished = {Web Page},
  Month = {Jul},
  Note = {Accessed: 2017-1-24},
  Year = 2015,
  Key = {www-buildstep},
  Url = {https://github.com/progrium/buildstep}
}
```

Is this entry correct? Let us analyse. But first we need to undersrand the semantics of the fields.

What are the Different entry Types and Fields

You see that the entry contains a number of fields. An extensive explanation of these fields can be found at

Please see <https://en.wikipedia.org/wiki/BibTeX>

We provide for this example a comprehensive discussions of the fields used. For other examples we suggest you refer to the document to identify what needs to be filled out.

Entry type Misc

First, it seems appropriate to use a *@misc* entry. We correctly identify this is a misc entry as it is online available. More recent version of bibtex include also the type *@online* for it. However, in order to maintain compatibility to older formats we chose simply Misc here and if we really would need to we could replace it easily.

Label

Typically the bibliography label should contain 3 letters from an author name, short year and the short name of the publication to provide maximum information regarding the publication. However in this case the project is hosted on github, so creating the label based on just the github location seems more logical. Thus our label that we propose is given by

```
github-progrium-buildstep
```

Under no circumstances should you use underscores as they can have unintended consequences in programs we use to create papers for our classes. Just use a minus sign instead. As it is hosted on github we also want the githubname and the projectname. As you can see we just derived it from the URL.

Indiana University

When managing bibliography entries with large numbers of collaborators it is advisable that all bibliography labels be initially be prefixed with an id for the collaborator. In our case we use the HID. Thus in our case we will have the following label.

```
hid-sp18-000-github-progrium-buildstep
```

Author

Normally we can write in the author fiels the names of the authors separated by the work and. IT is important tu use the word and but not use a comma. However, this only works if the lastname does not contain a space in it. In order to avoid confusing the system, we recommend therefore to write all names in the form Lastname, Firstname Middle Initials

Thus we find in our example

```
Author = {Gonzalez, Jose and Lindsay, Jeff},
```

Warning

Please note the word and between authors and not a comma. Commas are only used to distinguish between lastname and firstname.

Key

In this case the key field can be removed as the entry has an author field entry. If there was no author field, we use the key to specify the alphabetical ordering based on the specified key. Note that a key is not the label. In fact in our original entry the key field was wrongly used and the student did not understand that the key is used for sorting.

Howpublished

Since the source is a github project repository, the howpublished field shall hold the value {Code Repository}. If the url specified was a normal webpage, the {Web Page} entry would be valid.

Month

To allow internationalization of the month we use the first 3 letters in the english language for the month. Thus it is month = jul. Note that there are no brackets around the month.

Owner

In class we introduced the convention to put the student HID in it. If multiple students contributed, add them with space separation.

Accessed

As some styles do not support the accessed field, we simply include it in the note field. This is absolutely essential as code can change and when we read the code we looked at a particular snapshot in time. In addition it is often necessary to record the actual version of the code or the branch. Typically for github entries, it is best to just use the month and year field as some styles check for it.

Final Entry

Filling out as many fields as possible with information for this entry we get:

```
@Misc{github-progrium-buildstep,
  Title = {Buildstep},
  Author = {Jose Gonzalez and Jeff Lindsay},
  HowPublished = {Code Repository},
  Year = {2015},
  Month = jul,
  Note = {Accessed: 2017-1-24, master branch},
  Url = {https://github.com/progrium/buildstep},
  Owner = {S17-IO-3025},
}
```

We are using the release date in the year and month field as this project uses this for organizing releases. However, other project may have release versions so you would have in addition to using the data also to include the version in the note field such as:

```
Note = {Version: 1.2.3, Accessed: 2017-1-24},
```

12.2.2 Pedigree

Often it is advantageous to document the pedigree of the bibtex entries. Eg, how did you derive the final entry? To do so you can simply add a field such as `bibsource` and include in it all links you used to gather the final entry. However, we believe that the effort to curate an entry is sufficient to manage your own bibliography which you should freely distribute and constitute an own contribution.

12.2.3 Article in a Journal

Many online bibtex entries that you will find are wrong or incomplete. Often you may find via google a bibtex entry that may need some more research. Let us assume your first google query returns a publication and you cite it such as this:

```
@Unpublished{unpublished-google-sawzall,
  Title = {{Interpreting the Data: Parallel Analysis with Sawzall}},
  Author = {{Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan}},
  Note = {accessed 2017-01-28},
  Month = {October},
  Year = {2005},
  Owner = {for the purpose of this discussion removed},
  Timestamp = {2017.01.31}
}
```

Could we improve this entry to achieve your best? We observe:

1. The author field has a wrong entry as the `,` is to be replaced by an `and`.
2. The author field has authors and thus must not have a `{{ }}`
3. The url is missing, as the simple google search actually finds a PDF document.

Let us investigate a bit more while searching for the title. We find

- A https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwj_ytSA-PDRAhUH8IMKHaomC-oQFggaMAA&url=https%3A%2F%2Fresearch.google.com%2Farchive%2Fsawzall-sciprog.pdf&usg=AFQjCNHSSfKBwbxVAVPQ0td4rTjitKucpA&sig2=vbiVzi36B3gGFjIzlUKBDA&bvm=bv
- B <https://research.google.com/pubs/pub61.html>
- C <http://dl.acm.org/citation.cfm?id=1239658>

Let us look at (A)

As you can see from the url this is actually some redirection to a google web page which probably is replaced by B as its from google research. So let us look at (B)

Now when you look at the link we find the url <https://research.google.com/archive/sawzall-sciprogram.pdf> which redirects you to the PDF paper.

When we go to (B) we find surprisingly a bibtex entry as follows:

```
@article{61,
  title = {Interpreting the Data: Parallel Analysis with Sawzall},
  author = {Rob Pike and Sean Dorward and Robert Griesemer and Sean Quinlan},
  year = 2005,
  URL = {https://research.google.com/archive/sawzall.html},
  journal = {Scientific Programming Journal},
  pages = {277--298},
  volume = {13}
}
```

Now we could say let us be satisfied, but (C) seems to be even more interesting as its from a major publisher. So lets just make sure we look at (C)

If you go to (C), you find under the colored box entitled Tools and Resources a link called **bibtex**. Thus it seems a good idea to click on it. This will give you:

```
@article{Pike:2005:IDP:1239655.1239658,
  author = {Pike, Rob and Dorward, Sean and Griesemer, Robert and Quinlan, Sean},
  title = {Interpreting the Data: Parallel Analysis with Sawzall},
  journal = {Sci. Program.},
  issue_date = {October 2005},
  volume = {13},
  number = {4},
  month = oct,
  year = {2005},
  issn = {1058-9244},
  pages = {277--298},
  numpages = {22},
  url = {http://dx.doi.org/10.1155/2005/962135},
  doi = {10.1155/2005/962135},
  acmid = {1239658},
  publisher = {IOS Press},
  address = {Amsterdam, The Netherlands, The Netherlands},
}
```

Now we seem to be at a position to combine our search result as neither entry is sufficient. As the doi number properly specifies a paper (look up what a doi is) we can replace the url with one that we find online, such as the one we found in (A) Next we see that all field sin B are already covered in C, so we take (C) and add the url. Now as the label is great and uniform for ACM, but for us a bit less convenient as its difficult to remember, we just change it while for example using authors, title, and year information. Let us also make sure to do mostly lowercase in the label just as a convention. Thus our entry looks like:

```
@article{pike05swazall,
  author = {Pike, Rob and Dorward, Sean and Griesemer, Robert and Quinlan, Sean},
  title = {Interpreting the Data: Parallel Analysis with Sawzall},
  journal = {Sci. Program.},
  issue_date = {October 2005},
  volume = {13},
  number = {4},
  month = oct,
```

```

    year = {2005},
    issn = {1058-9244},
    pages = {277--298},
    numpages = {22},
    url = {https://research.google.com/archive/sawzall-sciprog.pdf},
    doi = {10.1155/2005/962135},
    acmid = {1239658},
    publisher = {IOS Press},
    address = {Amsterdam, The Netherlands, The Netherlands},
}

```

As you can see properly specifying a reference takes multiple google queries and merging of the results you find from various returns. As you still have time to correct things I advise that you check your references and correct them. If the original reference would have been graded it would have been graded with a ‘fail’ instead of a ‘pass’.

Naturally you need to judge if you can integrate the URL or not, often papers exist in a prepublication and you must make sure to cite the version you used. IN fact if you have the prepublicAtion, you should obtain the final manuscript as it could contain significant corrections to the previous draft. In other cases the prepublication may just be fine and you could for your own references keep the url. FOr the final publication you probably want to make sure that the doi is used. FOr the collection of your references, adding the url could be useful.

12.2.4 Article in a Conference Proceedings

Now let us look at another obvious example that needs improvement:

```

@InProceedings{wettinger-any2api,
  Title      = {Any2API - Automated APIfication},
  Author     = {Wettinger, Johannes and
               Uwe Breitenb{\u}cher
               and Frank Leymann},
  Booktitle  = {Proceedings of the 5th International
               Conference on Cloud Computing and
               Services Science},
  Year       = {2015},
  Pages      = {475-486},
  Publisher  = {SciTePress},
  ISSN       = {2326-7550},
  Owner      = {S17-IO-3005},
  Url        = {https://pdfs.semanticscholar.org/1cd4/4b87be8cf68ea5c4c642d38678a7b40a86de.pdf}
}

```

As you can see this entry seems to define all required fields, so we could be tempted to stop here. But its good to double check. Let us do some queries against ACM and google scholar. Let us just type in the title, and if this is in a proceedings they should return hopefully a predefined bibtex record for us.

Let us query:

```
google: googlescholar Any2API Automated APIfication
```

We get:

- https://scholar.google.de/citations?view_op=view_citation&hl=en&user=j6lIXtOAAAAJ&citation_for_view=j6lIXtOAAAAJ:8k81kl-MbHgC

On that page we see [Cite](#)

So we find a PDF at <https://pdfs.semanticscholar.org/1cd4/4b87be8cf68ea5c4c642d38678a7b40a86de>.

[pdf](#)

Let us click on this and the document includes a bibtex entry such as:

```
@inproceedings{Wettinger2015,
  author= {Johannes Wettinger and Uwe Breitenb{"u}cher and Frank
    Leymann},
  title = {Any2API - Automated APIfication},
  booktitle = {Proceedings of the 5th International Conference on Cloud
    Computing and Service Science (CLOSER)},
  year = {2015},
  pages = {475--486},
  publisher = {SciTePress}
}
```

Now let us add the URL and owner:

```
@inproceedings{Wettinger2015,
  author= {Johannes Wettinger and Uwe Breitenb{"u}cher and Frank
    Leymann},
  title = {Any2API - Automated APIfication},
  booktitle = {Proceedings of the 5th International Conference on Cloud
    Computing and Service Science (CLOSER)},
  year = {2015},
  pages = {475--486},
  publisher = {SciTePress},
  url ={https://pdfs.semanticscholar.org/1cd4/4b87be8cf68ea5c4c642d38678a7b40a86de.pdf},
  owner = {S17-IO-3005},
}
```

Should we be satisfied? No, even our original information we gathered provided more information. So let us continue. Let us issue additional google searches with ACM or IEEE and the title. When doing the IEEE in the example we find an entry called

[dlp: Frank Leyman](#)

Let us look at it and we find two entries:

```
@inproceedings{DBLP:conf/closer/WettingerBL15,
  author   = {Johannes Wettinger and
    Uwe Breitenb{"u}cher and
    Frank Leymann},
  title    = {{{ANY2API}} - Automated APIfication - Generating APIs for Executables
    to Ease their Integration and Orchestration for Cloud Application
    Deployment Automation},
  booktitle = {{{CLOSER}} 2015 - Proceedings of the 5th International Conference on
    Cloud Computing and Services Science, Lisbon, Portugal, 20-22 May,
    2015.},
  pages    = {475--486},
  year     = {2015},
  crossref = {DBLP:conf/closer/2015},
  url      = {http://dx.doi.org/10.5220/0005472704750486},
  doi      = {10.5220/0005472704750486},
  timestamp = {Tue, 04 Aug 2015 09:28:21 +0200},
  biburl   = {http://dblp.uni-trier.de/rec/bib/conf/closer/WettingerBL15},
  bibsource = {dblp computer science bibliography, http://dblp.org}
}

@proceedings{DBLP:conf/closer/2015,
  editor   = {Markus Helfert and
    Donald Ferguson and
    V{"i}ctor M{"e}ndez Mu{"n"}oz},
  title    = {{{CLOSER}} 2015 - Proceedings of the 5th International Conference on
```

```

        Cloud Computing and Services Science, Lisbon, Portugal, 20-22 May,
        2015}},
    publisher = {SciTePress},
    year      = {2015},
    isbn      = {978-989-758-104-5},
    timestamp = {Tue, 04 Aug 2015 09:17:34 +0200},
    biburl    = {http://dblp.uni-trier.de/rec/bib/conf/closer/2015},
    bibsource = {dblp computer science bibliography, http://dblp.org}
}

```

So let us look at the entry and see how to get a better one for our purpose and combine them. When using `jabref`, you see optional and required fields, we want to add as many as possible, regardless if optional or required, so Let us do that. We write it here in ASCII as it is easier to document and can also be done in `emacs`:

```

@InProceedings{
  author = {},
  title = {},
  OPTcrossref = {},
  OPTkey = {},
  OPTbooktitle = {},
  OPTyear = {},
  OPTeditor = {},
  OPTvolume = {},
  OPTnumber = {},
  OPTseries = {},
  OPTpages = {},
  OPTmonth = {},
  OPTaddress = {},
  OPTorganization = {},
  OPTpublisher = {},
  OPTnote = {},
  OPTannote = {},
  url = {}
}

```

Now we copy and fill out the **form** from our various searches:

```

@InProceedings{Wettinger2015any2api,
  author = {Johannes Wettinger and
    Uwe Breitenb{"u}cher and
    Frank Leymann},
  title = {{ANY2API - Automated APIfication - Generating APIs for Executables
    to Ease their Integration and Orchestration for Cloud Application
    Deployment Automation}},
  booktitle = {{CLOSER 2015 - Proceedings of the 5th International Conference on
    Cloud Computing and Services Science}},
  year = {2015},
  editor = {Markus Helfert and
    Donald Ferguson and
    V{"i"}ctor M{"e"}ndez Mu{-n}oz},
  publisher = {SciTePress},
  isbn = {978-989-758-104-5},
  pages = {475--486},
  month = {20-22 May},
  address = {Lisbon, Portugal},
  doi = {10.5220/0005472704750486},
  url = {https://pdfs.semanticscholar.org/1cd4/4b87be8cf68ea5c4c642d38678a7b40a86de.pdf},
  owner = {S17-I0-3005},
}

```

For the rest of the section we provide just some simple examples.

12.2.5 InProceedings

Please fill out

```

@InProceedings{
  author = {},
  title = {},
  OPTcrossref = {},
  OPTkey = {},
  OPTbooktitle = {},
  OPTyear = {},
  OPTeditor = {},
  OPTvolume = {},
  OPTnumber = {},
  OPTseries = {},
  OPTpages = {},
  OPTmonth = {},
  OPTaddress = {},
  OPTorganization = {},
  OPTpublisher = {},
  OPTnote = {},
  OPTannote = {},
  url = {}
}

@inproceedings{vonLaszewski15tas,
  author = {DeLeon, Robert L. and Furlani, Thomas R. and Gallo,
    Steven M. and White, Joseph P. and Jones, Matthew
    D. and Patra, Abani and Innus, Martins and Yearke,
    Thomas and Palmer, Jeffrey T. and Sperhac, Jeanette
    M. and Rathsam, Ryan and Simakov, Nikolay and von
    Laszewski, Gregor and Wang, Fugang},
  title = {{TAS View of XSEDE Users and Usage}},
  booktitle = {Proceedings of the 2015 XSEDE Conference: Scientific
    Advancements Enabled by Enhanced
    Cyberinfrastructure},
  series = {XSEDE '15},
  year = 2015,
  isbn = {978-1-4503-3720-5},
  location = {St. Louis, Missouri},
  pages = {21:1--21:8},
  articleno = 21,
  numpages = 8,
  url = {http://doi.acm.org/10.1145/2792745.2792766},
  doi = {10.1145/2792745.2792766},
  acmid = 2792766,
  publisher = {ACM},
  address = {New York, NY, USA},
  keywords = {HPC, SUPReMM, TAS, XDMoD, XSEDE usage, XSEDE users},
}

```

12.2.6 TechReport

Please fill out

```

@TechReport{
  author = {},
  title = {},
  institution = {},
  year = {},
  OPTkey = {},
}

```



```

OPTtype = {},
OPTnumber = {},
OPTaddress = {},
OPTmonth = {},
OPTnote = {},
OPTannote = {},
url = {}
}

@TechReport{las05exp,
  title = {{The Java CoG Kit Experiment Manager}},
  Author = {von Laszewski, Gregor},
  Institution = {Argonne National Laboratory},
  Year = 2005,
  Month = jun,
  Number = {P1259},
  url = {https://laszewski.github.io/papers/vonLaszewski-exp.pdf}
}

```

12.2.7 Article

Please fill out

```

@Article{,
  author = {},
  title = {},
  journal = {},
  year = {},
  OPTkey = {},
  OPTvolume = {},
  OPTnumber = {},
  OPTpages = {},
  OPTmonth = {},
  OPTnote = {},
  OPTannote = {},
  url = {}
}

@Article{las05gridhistory,
  title = {{The Grid-Idea and Its Evolution}},
  author = {von Laszewski, Gregor},
  journal = {Journal of Information Technology},
  year = 2005,
  month = jun,
  number = 6,
  pages = {319-329},
  volume = 47,
  doi = {10.1524/itit.2005.47.6.319},
  url = {https://laszewski.github.io/papers/vonLaszewski-grid-idea.pdf}
}

```

12.2.8 Proceedings

Please fill out

```

@Proceedings{,
  title = {},
  year = {},
  OPTkey = {},
  OPTbooktitle = {},
  OPTeditor = {},

```

```

OPTvolume = {},
OPTnumber = {},
OPTseries = {},
OPTaddress = {},
OPTmonth = {},
OPTorganization = {},
OPTpublisher = {},
OPTnote = {},
OPTannotate = {},
url = {}
}

@Proceedings{las12fedcloud-proc,
  title = {{FederatedClouds '12: Proceedings of the 2012
           Workshop on Cloud Services, Federation, and the 8th
           Open Cirrus Summit}},
  year = 2012,
  address = {New York, NY, USA},
  editor = {vonLaszewski, Gregor and Robert Grossman and Michael
           Kozuchand Rick McGeerand Dejan Milojicic},
  publisher = {ACM},
  isbn = {978-1-4503-1754-2},
  location = {San Jose, California, USA},
  url =
           {http://dl.acm.org/citation.cfm?id=2378975&picked=prox&cfid=389635474&cftoken=32712}
}

```

12.2.9 Wikipedia Entry

Please fill out

```

@Misc{,
  OPTkey = {},
  OPTauthor = {},
  OPTtitle = {},
  OPThowpublished = {},
  OPTmonth = {},
  OPTyear = {},
  OPTnote = {},
  OPTannotate = {},
  url = {}
}

@Misc{www-ode-wikipedia,
  Title = {Apache ODE},
  HowPublished = {Web Page},
  Note = {Accessed: 2017-2-11},
  Key = {Apache ODE},
  Url = {https://en.wikipedia.org/wiki/Apache_ODE}
}

```

12.2.10 Blogs

Please fill out

```

@Misc{,
  OPTkey = {},
  OPTauthor = {},
  OPTtitle = {},
  OPThowpublished = {},
  OPTmonth = {},

```

```

    OPTyear = {},
    OPTnote = {},
    OPTannote = {},
    OPTurl = {}
}

@Misc{www-clarridge-discoproject-blog,
  title = {Disco - A Powerful Erlang and Python Map/Reduce
           Framework},
  uthor = {Clarridge, Tait},
  howpublished = {Blog},
  month = may,
  note = {Accessed: 25-feb-2017},
  year = 2014,
  url = {http://www.taitclarridge.com/techlog/2014/05/disco-a-powerful-erlang-and-python-mapreduce-f
}

```

12.2.11 Web Page

Please fill out

```

@Misc{,
  OPTkey = {},
  OPTauthor = {},
  OPTtitle = {},
  OPThowpublished = {},
  OPTmonth = {},
  OPTyear = {},
  OPTnote = {},
  OPTannote = {},
  url = {}
}

@Misc{www-cloudmesh-classes,
  OPTkey = {},
  author = {von Laszewski, Gregor},
  title = {Cloudmesh Classes},
  howpublished = {Web Page},
  OPTmonth = {},
  OPTyear = {},
  OPTnote = {},
  OPTannote = {},
  url = {https://cloudmesh.github.io/classes/}
}

@Misc{www-awslambda,
  title = {AWS Lambda},
  author = {{Amazon}},
  key = {AWS Lambda},
  howpublished = {Web Page},
  url = {https://aws.amazon.com/lambda/faqs/}
}

```

12.2.12 Book

Given the following entry. What is the proper entry for this book. Provide rationale:

```

@Book{netty-book,
  Title = {Netty in Action},
  Author = {Maurer, Norman and Wolfthal, Marvin},
  Publisher = {Manning Publications},
}

```

```

    Year = {2016},
}

```

To obtain the record of a book you can look at many information sources. The can include:

- <https://www.manning.com/books/netty-in-action>
- <https://www.amazon.com/Netty-Action-Norman-Maurer/dp/1617291471>
- <http://www.barnesandnoble.com/w/netty-in-action-norman-maurer/1117342155?ean=9781617291470#productInfoTabs>
- <http://www.powells.com/book/netty-in-action-9781617291470/1-0>

Furthermore, we need to consider the entry of a book, we simply look it up in emacs where we find the following but add the owner and the url field:

```

@Book{,
  ALTAuthor = {},
  ALTeditor = {},
  title = {},
  publisher = {},
  year = {},
  OPTkey = {},
  OPTvolume = {},
  OPTnumber = {},
  OPTseries = {},
  OPTaddress = {},
  OPTedition = {},
  OPTmonth = {},
  OPTnote = {},
  OPTannote = {},
  owner = {},
  url = {}
}

```

In summary we find the following fields:

Required fields: author/editor, title, publisher, year

Optional fields: volume/number, series, address, edition, month, note, key

We apply the following to fill out the fields which is the standard definition as defined by L^AT_EX.

address: The address is the Publisher’s address. Usually just the city, but can be the full address for lesser-known publishers.

author: The name(s) of the author(s) (in the case of more than one author, separated by and) Names can be written in one of two forms: Donald E. Knuth or Knuth, Donald E. or van Halen, Eddie. Please note that Eddie van Halen would result in a wrong name. For our purpose we keep nobelity titles part of the last name.

edition: The edition of a book, long form (such as ‘First’ or ‘Second’)

editor: The name(s) of the editor(s)

key: A hidden field used for specifying or overriding the alphabetical order of entries (when the ‘author’ and ‘editor’ fields are missing). Note that this is very different from the key that is used to cite or cross-reference the entry.

label: The label field should contain three letters from the auth field, a short year reference and a short name of the publication to provide the maximum information regarding the publication. Underscores should be replaced with dashes or removed completely.

month: The month of publication or, if unpublished, the month of creation. Use three-letter abbreviations for this field in order to account for languages that do not capitalize month names. Additional information for the day can be included as follows: aug#~ ‘ ‘10, ’ ’

publisher: The publisher's name

series: The series of books the book was published in (e.g. "The Hardy Boys" or "Lecture Notes in Computer Science")

title: The title of the work. As the capitalization depends on the bibliography style and the language used we typically use camel case. To force capitalization of a word or its first letter you can use the curly braces, '{ }'. To keep the title in camel case simple use title = {{My Title}}

type: The field overriding the default type of publication (e.g. "Research Note" for techreport, "{PhD} dissertation" for phdthesis, "Section" for inbook/incollection) volume The volume of a journal or multi-volume book year The year of publication (or, if unpublished, the year of creation)

While applying the above rules and tips we summarize what we have done for this entry:

1. Search for the book by title/Author on ACM (<http://dl.acm.org/>) or Amazon or barnesandnoble or upcitemdb (<http://upcitemdb.com>). These services return bibtex entrie that you can improve.
2. Hence one option is t get the ISBN of the book. For "Mesos in action" from upcitemdb we got the ISBN as "9781617 292927". This is the 13 digit ISBN. The first 3 digits (GS1 code) can be skipped. Using the rest of 10 digits "1617 292927", Add in JabRef in Optional Fields->ISBN.

However it is fine to youst specify the full number.

We can also return a bibtex entry generated while using Click on the "Get BibTex from ISBN".

Now we get more information on this book entry from ISBN. We can opt either the original or newly searched entry for the below bibtex fields or merge as appropriate. URL may not match from where we initially read the book, however there is option to put your original url or newly searched url. EAN, Edition, Pages,url,published date etc. Do a search on amazon for "ASIN". Can skip if not available. Sometime we get ASIN for a different publication, maybe a paperback ASIN={B01MT311CU} We can add it as it becomes easier to search

doi: If you can find a doi numer you should also add it. IN this case we could not locate one.

As a result we obtain the entry:

```
@Book{netty-book,
  title = {Netty in Action},
  publisher = {Manning Publications Co.},
  year = {2015},
  author = {Maurer, Norman and Wolfthal, Marvin Allen},
  address = {Greenwich, CT, USA},
  edition = {1st},
  isbn = {1617291471},
  asin = {1617291471},
  date = {2015-12-23},
  ean = {9781617291470},
  owner = {S17-I0-3022 S17-I0-3010 S17-I0-3012},
  pages = {296},
  url = {http://www.ebook.de/de/product/21687528/norman_maurer_netty_in_action.html},
}
```

12.3 Integrating Bibtex entries into Other Systems

We have not tested any of this

12.3.1 jabref and MSWord

According to others it is possible to integrate jabref references directly into MSWord. For more information please see:

- <https://www.paulkiddie.com/2009/07/jabref-exports-to-word-2007-xml/>

12.3.2 Bibtex import to MSWord

XML import

Please give feedback if you used this.

see:

- <http://blog.pengyifan.com/using-bibtex-in-ms-word-2015-mac-os/>

1. In JabRef, export the bibliography in MS Word 2008 xml format
2. Name the file Sources.xml (case sensitive)
3. In OSX with MS Word 2015: Go to `/Library/Containers/com.microsoft.word/Data/Library/Applica`
4. Rename the original Sources.xml file to Sources.xml.bak
5. Copy the generated Sources.xml in this folder
6. Restart MS Word.

We do not know what needs to be done in case you need to make changes to the references. Please report back your experiences. To avoid issues we recommend that you use \LaTeX and not MSWord.

BibTex4Word

We have not tried this:

- <http://www.ee.ic.ac.uk/hp/staff/dmb/perl/index.html>

You are highly recommended to use Jabref for bibliography management in this class. Here is an introductory video on Jabref: <https://youtu.be/roi7vezNmfo?t=8m6s>

12.4 Other Reference Managers

Please note that you should first decide which reference manager you like to use. In case you for example install zotero and mendeley, that may not work with word or other programs.

12.4.1 Endnote

Endnote is a reference manager that works with Windows. Many people use Endnote. However, in the past, Endnote has caused complications when dealing with collaborative management of references. Its price is considerable. We have lost many hours of work because of instability of Endnote in some cases. As a student, you may be able to use Endnote for free at Indiana University.

- <http://endnote.com/>

12.4.2 Mendeley

Mendeley is a free reference manager compatible with Windows Word 2013, Mac Word 2011, LibreOffice, BibTeX. Videos on how to use it are available at:

- <https://community.mendeley.com/guides/videos>

Installation instructions are available at

- <https://www.mendeley.com/features/reference-manager/>

When dealing with large databases, we found the integration of Mendeley into word slow.

12.4.3 Zotero

Zotero is a free tool to help you collect, organize, cite, and share your research sources. Documentation is available at

- <https://www.zotero.org/support/>

The download link is available from

- <https://www.zotero.org/>

We have limited experience with Zotero

12.4.4 Paperpile

Paper pile is a Web based reference management tool that integrates with Google docs. It can export the database as bibtex. Paperpile is a commercial tool costing about \$36 a year for academic users. For others it is about 3 times as expensive.

- <https://paperpile.com>



13. Editors

13.1 Basic Emacs

One of the most useful short manuals for emacs is the following reference card. It takes some time to use this card efficiently, but the most important commands are written on it. Generations of students have literally been just presented with this card and they learned emacs from it.

- <https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf>

There is naturally also additional material available and a great manual. You could also look at

- <https://www.gnu.org/software/emacs/tour/>

From the last page we have summarized the most useful and **simple** features. And present them here. One of the hidden gems of emacs is the ability to recreate replay able macros which we include here also. You ought to try it and you will find that for data science and the cleanup of data emacs (applied to smaller datasets) is a gem.

Notation

Key	Description
C	Control
M	Esc (meta character)

Here are some other ways on what to do if you have accidentally pressed a wrong key:

- C-g If you pressed a prefix key (e.g. C-x) or you invoked a command which is now prompting you for input (e.g. Find file: ...), type C-g, repeatedly if necessary, to cancel. C-g also cancels a long-running operation if it appears that Emacs has frozen.

- `C-/` If you executed a command and Emacs has modified your buffer, use `C-/` to undo that change.

To save the current file say

Key	Description
<code>C-x C-w</code>	Write the buffer to file
<code>C-x C-s</code>	Write the buffer to file and quit Emacs

Moving around in buffers can be done with cursor keys, or with the following key combinations:

Key	Description
<code>C-f</code>	Forward one character
<code>C-n</code>	Next line
<code>C-b</code>	Back one character
<code>C-p</code>	Previous line

Here are some ways to move around in larger increments:

Key	Description
<code>C-a</code>	Beginning of line
<code>M-f</code>	Forward one word
<code>M-a</code>	Previous sentence
<code>M-v</code>	Previous screen
<code>M-<</code>	Beginning of buffer
<code>C-e</code>	End of line
<code>M-b</code>	Back one word
<code>M-e</code>	Next sentence
<code>C-v</code>	Next screen
<code>M-></code>	End of buffer

You can jump directly to a particular line number in a buffer:

Key	Description
<code>M-g g</code>	Jump to specified line

Searching is easy with the following commands

Key	Description
<code>C-s</code>	Incremental search forward
<code>C-r</code>	Incremental search backward

Replace

Key	Description
M-%	Query replace

Killing (“cutting”) text

Key	Description
C-k	Kill line

Yanking

Key	Description
C-y	Yanks last killed text

Macros

Keyboard Macros

Keyboard macros are a way to remember a fixed sequence of keys for later repetition. They’re handy for automating some boring editing tasks.

Key	Description
M-x (Start recording macro
M-x)	Stop recording macro
M-x e	Play back macro once
M-5 C-x-e	Play back macro 5 times

Modes


“Every buffer has an associated major mode, which alters certain behaviors, key bindings, and text display in that buffer. The idea is to customize the appearance and features available based on the contents of the buffer.” modes are typically activated by ending such as `.py`, `.java`, `.rst`, ...

Key	Description
M-x <code>python-mode</code>	Mode for editing Python files
M-x <code>auto-fill-mode</code>	Wraps your lines automatically when they get longer than 70 characters.
M-x <code>flyspell-mode</code>	Highlights misspelled words as you type.

13.1.1 Org Mode

Emacs has some very advanced features that you can activate via a mode. One such feature is to organize a TODO list via `org-mode`.

Instead of us designing our own video, we point to a community tutorial such as

[Emacs org-mode \(18:04\)](#)  Youtube

13.1.2 Programming Python with Emacs

Emacs comes by default with syntax highlighting for python when you edit a .py file. This is really all you need. It also comes with a python ide that you can use and customize.

Python auto-completion for Emacs:

- <https://github.com/tkf/emacs-jedi>

Some more information is available at

- <https://realpython.com/blog/python/emacs-the-best-python-editor/>
- <https://www.emacswiki.org/emacs/PythonProgrammingInEmacs>

13.1.3 Emacs Keys in a Terminal

One of the real great features of knowing emacs is that you can set all your editors to emacs shortcuts. This includes pyCharm, but also bash. IN bash you simply say

```
set -o emacs
```

in your bash prompt. Additionally, if you do not have a window systems configured, you can run emacs directly in the terminal with

```
emacs -nw
```

This you can log in to a remote computer and if it has emacs installed. Use it in the terminal. This would replace editors such as vi, vim, nano, pico or others that work in a terminal.

13.1.4 \LaTeX and Emacs

LaTeX is directly supported by emacs and nothing has to be changed. However, a collection of information about additional \LaTeX features for emacs is available at

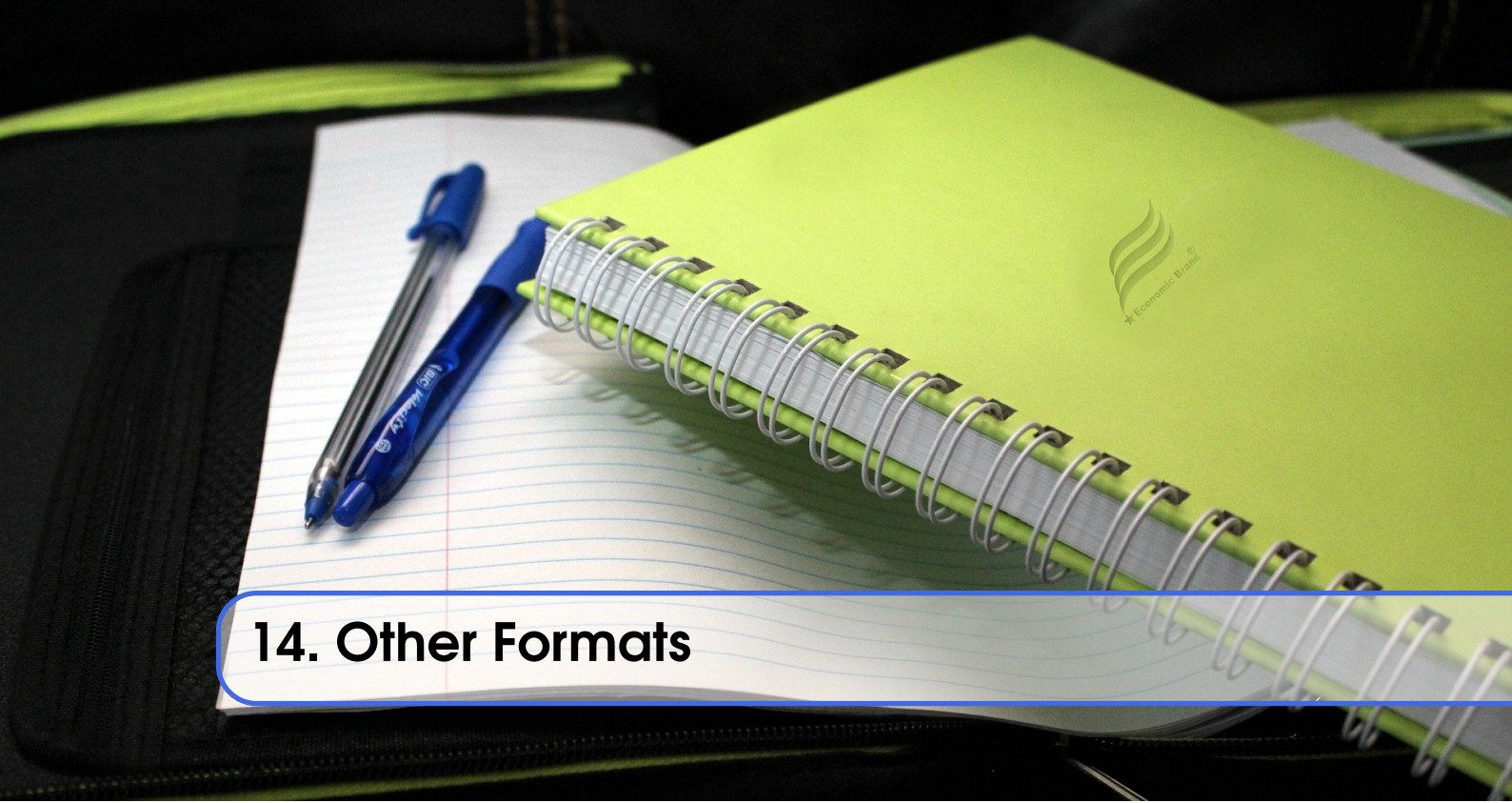
- <https://www.emacswiki.org/emacs/LaTeX>

Of interest are for example also

- * M-x flyspell-mode: allowing to do spell checking in the window
- * predictive mode: <https://www.emacswiki.org/em>
- * preview latex
- * whizzy tex

However instead of previes and whizzy tex we recommend to use <https://www.emacswiki.org/emacs/LatexMk> which comes preinstalled and allows you to do editing in one terminal, while previewing the update on change in another window.

LatexMk is all integrated in our report Makefiles and the Book format, so you will be able to use this immediately. This is similar to share latex, but much faster and without collaborators editing the same file.



14. Other Formats

14.1 reStructuredText

reStructuredText (RST) purpose is to provide an easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system. With its help you can develop documentation not only for stand alone documentation, simple web pages, an in-line program documentation (such as Python). RST is extensible and new features can be added. It is used in sphinx as one of its supported formats.

14.1.1 Links

- RST Sphinx documentation: <http://www.sphinx-doc.org/en/stable/rest.html>
- RST Syntax: <http://docutils.sourceforge.net/rst.html>
- Important extensions: <http://sphinx-doc.org/ext/todo.html>

Cheatcheat:

- <http://github.com/ralsina/rst-cheatsheet/raw/master/rst-cheatsheet.pdf>
- <http://docutils.sourceforge.net/docs/ref/rst/directives.html>

14.1.2 Source

The source for this page is located at

- <https://raw.githubusercontent.com/cloudmesh/classes/master/docs/source/lesson/doc/rst.rst>

This way you can look at the source on how we create this page.

14.1.3 Sections

with overline, for parts * with overline, for chapters =, for sections -, for subsections ^, for subsubsections ", for paragraphs

RST allows to specify a number of sections. You can do this with the various underlines:

```
*****
Chapter
*****
Section
=====
Subsection
-----
Subsubsection
^^^^^^^^^^^^^^^^
Paragraph
~~~~~
```

14.1.4 Listtable

```
.. csv-table:: Eye colors
   :header: "Name", "Firstname", "eyes"
   :widths: 20, 20, 10

   "von Laszewski", "Gregor", "gray"
```

14.1.5 Excelltable

we have integrated Excel table from <http://pythonhosted.org//sphinxcontrib-exceltable/> into our sphinx allowing the definition of more elaborate tables specified in excel. However the most convenient way may be to use list-tables. The documentation to list tables can be found at <http://docutils.sourceforge.net/docs/ref/rst/directives.html#list-table>

14.1.6 Boxes

Seealso

```
.. seealso:: This is a simple seealso note.
```

Note

This is a **note** box.

```
.. note:: This is a note box.
```

Warning

note the space between the directive and the text

```
.. warning:: note the space between the directive and the text
```

Others

This is an **attention** box.

```
.. attention:: This is an attention box.
```

This is a **caution** box.

```
.. caution:: This is a caution box.
```

This is a **danger** box.

```
.. danger:: This is a danger box.
```

This is a **error** box.

```
.. error:: This is a error box.
```

This is a **hint** box.

```
.. hint:: This is a hint box.
```

This is an **important** box.

```
.. important:: This is an important box.
```

This is a **tip** box.

```
.. tip:: This is a tip box.
```

14.1.7 Sidebar directive

It is possible to create sidebar using the following code:

```
.. sidebar:: Sidebar Title
   :subtitle: Optional Sidebar Subtitle
```

```
Subsequent indented lines comprise
the body of the sidebar, and are
interpreted as body elements.
```

Sidebar Title: Optional Sidebar Subtitle

Subsequent indented lines comprise the body of the sidebar, and are interpreted as body elements.

14.1.8 Sphinx Prompt

```
.. prompt:: bash, cloudmesh$

wget -O cm-setup.sh http://bit.ly/cloudmesh-client-xenial
sh cm-setup.sh
```

14.1.9 Programm examples

You can include code examples and bash commands with two colons.

This is an example for python:

```
print ("Hallo World")
```

This is an example for a shell command:

```
$ ls -lisa
```

14.1.10 Hyperlinks

Direct links to html pages can ve done with:

```
'This is a link to an html page <hadoop.html>'
```

Note that this page could be generated from an rst page

Links to the FG portal need to be formulated with the portal tag:

```
:portal:'List to FG projects </projects/all>'
```

In case a subsection has a link declared you can use `:ref:`. This is the preferred way as it can be used to point even to subsections:

```
:ref:'Connecting private network VMs clusters <_s_vpn>'
```

A html link can be created anywhere in the document but must be unique. for example if you place:

```
.. _s_vpn:
```

in the text it will create a target to which the above link points when you click on it

14.1.11 Todo

```
.. todo:: an example
```

Todo
an example

14.2 Markdown

The content from this section originates from see: <https://en.wikipedia.org/wiki/Markdown>.

Markdown is a simple markup language, however there is no precise standard defined for it and implementations may have features not supported by other implementations. Nevertheless, it provides as simple and easy way to quickly develop clean looking documents.

There are several tools that make markdown attractive allowing to write the text in one window while at the same time seeing the rendered output in another.

This includes

Macdown An editor for markdown targeted on OSX

To convert the markdown to other formats with pandoc

```
# Heading
```

```
## Sub-heading
```

```
### Another deeper heading
```

```
Paragraphs are separated  
by a blank line.
```

```
Two spaces at the end of a line leave a  
line break.
```

```
Text attributes _italic_, *italic*, __bold__, **bold**, 'monospace'.
```

```
Horizontal rule:
```

```
---
```

```
Bullet list:
```


- * apples
- * oranges
- * pears

Numbered list:

1. apples
2. oranges
3. pears

A [link] (<http://example.com>).

14.2.1 Tools

Dilinger • <https://dillinger.io/>

. A HTML5 based cloud enabled editor. It allows to download the created Markdown.

Macdown • <https://macdown.uranusjr.com/>

“MacDown is an open source Markdown editor for macOS”

14.2.2 Presentations in Markdown

Please find some links on how to use markdown to create slides

<https://yhatt.github.io/marp/>

<http://slidify.org/>

<https://rmarkdown.rstudio.com/lesson-11.html>

GitPitch <https://github.com/gitpitch/gitpitch/wiki/Slide-Markdown>

14.3 Communicating Research in Other Ways

Naturally, writing papers is not the only way to communicate your research with others. We find that today we see additional pathways for communication including blogs, twitter, facebook, e-mail, Web pages, and electronic notebooks. Let us revisit some of them and identify when they are helpful.

14.3.1 Blogs

blog: noun, a regularly updated website or web page, typically one run by an individual or small group, that is written in an informal or conversational style.

Advantages:

- encourages spontaneous posts
- encourages small short contributions
- chronologically ordered
- standard software exists to set up blogs
- online services exist to set up blogs

Disadvantages:

- structuring data is difficult (some blog software support it)
- not suitable for formal development of a paper

- often lack of sophisticated track change features
- no collaborative editing features

14.3.2 Sphinx

Sphinx (<http://www.sphinx-doc.org/>) is a tool that to create integrated documentation from a markup language whlie.

Advantages:

- output formats: html, LaTeX, PDF, ePub
- integrates well with directory structure
- powerful markup language (reStructuredText)
- can be hosted on github via github pages
- can integare other renderers such as Markdown
- automatic table of content, tebale of index
- code documentation integration
- search
- written in python and using bash, so extensions and custom automation are possible

Disadvantage:

- requires compile step
- When using markdown github can render individual page

Others:

- Read the Docs (<https://readthedocs.org/>)
- Doxygen (<http://www.stack.nl/~dimitri/doxygen/>)
- MkDocs (<http://www.mkdocs.org/>)

14.3.3 Notebooks

Jupyter

The Jupyter Notebook (<http://jupyter.org/>) is an open-source web application allowing users to create and share documents that contain live code, equations, visualizations and explanatory text. Use cases include data cleaning and transformation, numerical simulation, statistical modeling, machine learning.

Advantages:

- Integrates with python
- Recently other programming languages have been integrated
- Allows experimenting with settings
- Allows a form of literate programming while mixing documentation with code
- automatically renders on github
- comes with web service that allows hosting

Disadvantage:

- mostly encourages short documents
- mark up language is limited
- editing in ASCII is complex and Web editing is prefered

Apache Zeppelin

A Web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala and hadoop. It integrates a web-based notebook with data ingestion, data exploration, visualization, sharing and collaboration features to Hadoop and Spark.

Advantages:

- integration to various framework
- Web framework
- integration with spark, hadoop

Disadvantages:

- larger framework
- must leverages existing deployments of spak, hadoop



Development Tools

15	Linux	209
15.1	History	
15.2	Shell	
15.3	Multi-command execution	
15.4	Keyboard Shortcuts	
15.5	bashrc and bash_profile	
15.6	Makefile	
15.7	Exercises	
16	SSH	215
16.1	Generate a SSH key	
16.2	Add or Replace Passphrase	
16.3	SSH Add and Agent	
16.4	SSH Config	
16.5	SSH Port Forwarding	
16.6	Upload the key to gitlab	
16.7	Using SSH on Mac OS X	
16.8	Using SSH on Linux	
16.9	Using SSH on Raspberry Pi 3	
16.10	SSH on Windows	
16.11	Tips	
16.12	SSH to FutureSystems Resources	
16.13	Testing your ssh key	
16.14	Refernces	
16.15	Exercises	
17	Github	225
17.1	Obverview	
17.2	Upload Key	
17.3	Fork	
17.4	Rebase	
17.5	Remote	
17.6	Pull Request	
17.7	Branch	
17.8	Checkout	
17.9	Merge	
17.10	GUI	
17.11	Windows	
17.12	Git from the Commandline	
17.13	Configuration	
17.14	Upload your public key	
17.15	Working with a directory that was provided for you	
17.16	README.yml and notebook.md	
17.17	Contributing to the Document	
17.18	Exercises	
17.19	Github Issues	
18	Virtual Box	235
18.1	Instalation	
18.2	Guest additions	
18.3	Exercises	



15. Linux

15.1 History

LINUX is a reimplementation by the community of UNIX which was developed in 1969 by Ken Thompson and Dennis Ritchie of Bell Laboratories and rewritten in C. An important part of UNIX is what is called the *kernel* which allows the software to talk to the hardware and utilize it.

In 1991 Linus Trovalds started developing a Linux Kernel that was initially targeted for PC's. This made it possible to run it on Laptops and was later on further developed by making it a full Operating system replacement for UNIX.

15.2 Shell

One of the most important features for us will be to access the computer with the help of a *shell*. The shell is typically run in what is called a terminal and allows interaction to the computer with commandline programs.

There are many good tutorials out there that explain why one needs a linux shell and not just a GUI. Randomly we picked the first one that came up with a google query. This is not an endorsement for the material we point to, but could be a worth while read for someone that has no experience in Shell programming:

- http://linuxcommand.org/lc3_learning_the_shell.php

Certainly you are welcome to use other resources that may suite you best. We will however summarize in table form a number of useful commands that you may als find even as a RefCard.

- <http://www.cheat-sheets.org/#Linux>

Table 15.1 -- continued from previous page

Example	Description
uname date time <i>command</i> shutdown -h ‘shut down’	(assignment) prints the current date and time prints the sys, real and user time (assignment)
Networking Commands	
ping netstat hostname traceroute ifconfig	(assignment) (assignment) (assignment) (assignment) (assignment)
Internet Commands	
host whois dig wget curl	(assignment) (assignment) (assignment) (assignment) (assignment)
Remote Access Commands	
ssh scp sftp	(assignment) (assignment) (assignment)

15.3 Multi-command execution

One of the important features is that one can execute multiple commands in the shell.

To execute command 2 once command 1 has finished use

```
command1; command2
```

To execute command 2 as soon as command 1 forwards output to stdout use

```
command1; command2
```

To execute command 1 in the background use

```
command1 &
```

15.4 Keyboard Shortcuts

These shortcuts will come in handy. Note that many overlap with emacs short cuts.

Keys	Description
Up Arrow	Show the previous command
Ctrl + z	Stops the current command Resume with fg in the foreground Resume with bg in the background
Ctrl + c	Halts the current command
Ctrl + l	Clear the screen
Ctrl + a	Return to the start of the line
Ctrl + e	Go to the end of the line
Ctrl + k	Cut everything after the cursor to a special clipboard
Ctrl + y	Paste from the special clipboard
Ctrl + d	Logout of current session, similar to exit

15.5 bashrc and bash_profile

Usage of a particular command and all the attributes associated with it, use ‘man’ command. Avoid using `rm -r` command to delete files recursively. A good way to avoid accidental deletion is to include the following in your `.bash_profile` file:

```
alias e=open_emacs
alias rm='rm -i'
alias mv='mv -i'
alias h='history'
```

More Information

<https://cloudmesh.github.io/classes/lesson/linux/refcards.html>

15.6 Makefile

Makefiles allow developers to coordinate the execution of code compilations. This not only includes C or C++ code, but any translation from source to a final format. For us this could include the creation of PDF files from latex sources, creation of docker images, and the creation of cloud services and their deployment through simple workflows represented in makefiles, or the coordination of execution targets.

As makefiles include a simple syntax allowing structural dependencies they can easily adapted to fulfill simple activities to be executed in repeated fashion by developers.

An example of how to use Makefiles for docker is provided at <http://jmkhael.io/makefiles-for-your-dockerfiles/>.

An example on how to use Makefiles for L^AT_EX is provided at <https://github.com/cloudmesh/book/blob/master/Makefile>.

Makefiles include a number of rules that are defined by a target name. Let us define a target called `hello` that prints out the string “Hello World”.

```
1 hello:
2   @echo "Hello World"
```

Important to remember is that the commands after a target are not indented just by spaces, but actually by a single TAB character. Editors such as emacs will be ideal to edit such Makefiles, while allowing syntax highlighting and easy manipulation of TABs. Naturally other editros will do

that also. Please choose your editor of choice. One of the best features of targets is that they can depend on other targets. Thus, if we define

```
1 hello: hello
2   @echo "Hallo World"
```

our makefile will first execute hello and then all commands in hello. As you can see this can be very useful for defining simple dependencies.

In addition we can define variables in a makefile such as

```
1 HELLO="Hello World"
2
3 hello:
4   @echo $(HELLO)
```

and can use them in our text with \$ invocations.

Moreover, in sophisticated Makefiles, we could even make the targets dependent on files and a target rule could be defined that only compiles those files that have changed since our last invocation of the Makefile, saving potentially a lot of time. However, for our work here we just use the most elementary makefiles.

For more information we recommend you to find out about it on the internet. A convenient reference card is available at <http://www.cs.jhu.edu/~joanne/unixRC.pdf>.

Makefiles on Windows

Makefiles can easily be accessed also on windows while installing gitbash. Please refer to the internet or search in this handbook for more information about gitbash.

15.7 Exercises


Exercise 15.1 Familiarize yourself with the commands ■

Exercise 15.2 Find more commands that you find useful and add them to this page. ■

Exercise 15.3 Use the sort command to sort all lines of a file while removing duplicates. ■

Exercise 15.4 In Table 15.1 you will find a number of commands with (assignment). Develop descriptions that you will contribute and add to the manual with a pull request. Work in a team so that only one pull request is issued. Do not only provide the description, but also a real example as showcased within the table. ■

Exercise 15.5 Should there be other commands listed in the table. If so which? Create a pull request for them. ■



16. SSH

If you do not know what ssh is we recommend that you [read up on it](#). However, the simple material presented here will help you get started quickly. It can however not replace the more comprehensive documentation. We also recommend that you read this entire section first before you start past and copy style tutorials as this will not allow you to understand the way ssh is used. For this reason we start with the explanation of ssh and not the setup on your machine. We want you to obtain first a solid overview of what you need to do.

To access remote resources this is often achieved via SSH. You need to provide a public ssh key to FutureSystem. We explain how to generate a ssh key, upload it to the FutureSystem portal and log onto the resources. This manual covers UNIX, Mac OS X, as well as Windows 10.

16.1 Generate a SSH key

First we must generate a ssh key with the tool [ssh-keygen](#). This program is commonly available on most UNIX systems (this includes Cygwin if you installed the ssh module or use our pre-generated cygwin executable). It will ask you for the location and name of the new key. It will also ask you for a passphrase, which you **MUST** provide. Please use a strong passphrase to protect it appropriately.

Warning

Some teachers and teaching assistants advise you to not use passphrases. This is **WRONG** as it allows someone that gains access to your computer to also gain access to all resources that have the public key.

In case you already have a ssh key in your machine, you can reuse it and skip this whole section.

To generate the key, please type:

Example:

```
ssh-keygen -t rsa -C localname@indiana.edu
```

This command requires the interaction of the user. The first question is:

```
Enter file in which to save the key (/home/localname/.ssh/id_rsa):
```

We recommend using the default location `~/.ssh/` and the default name `id_rsa`. To do so, just press the enter key.

Your *localname* is the username on your computer.

The second and third question is to protect your ssh key with a passphrase. This passphrase will protect your key because you need to type it when you want to use it. Thus, you can either type a passphrase or press enter to leave it without passphrase. To avoid security problems, you **MUST** chose a passphrase. Make sure to not just type return for an empty passphrase:

```
Enter passphrase (empty for no passphrase):
```

and:

```
Enter same passphrase again:
```

If executed correctly, you will see some output similar to:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/localname/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/localname/.ssh/id_rsa.
Your public key has been saved in /home/localname/.ssh/id_rsa.pub.
The key fingerprint is:
34:87:67:ea:c2:49:ee:c2:81:d2:10:84:b1:3e:05:59 localname@indiana.edu

+--[ RSA 2048 ]-----+
|.+. . . . .Eo= .      |
|..=.o + o +o        |
|0. = .....          |
| = . . .             |
+-----+

```

Once, you have generated your key, you should have them in the `.ssh` directory. You can check it by:

```
$ cat ~/.ssh/id_rsa.pub
```

If everything is normal, you will see something like:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACXJH2iG2FMHqC6T/U7uB8kt
6KlRh4kU0jgw9sc4Uu+Uwe/kshuispauhfsjhfm,anf6787sjgdkjsgl+EwD0
thkoamyi0VvhTVZhj61pTdhy11t8hlkoL19JVnVBPP5kIN3wVyNAJjYBrAUNW
4dXKXtmfkXp98T30W4mxAtTH434MaT+QcPTcxims/hwsUeDAVKZY7UgZhEbiE
xxkejtnRBHTipi0W03W05TOUGRW7EuKf/4ftNVPi1C04DpfY44NFG1xPwHeim
Uk+t9h48pBQj16FrUCpOrS02Pj+4/9dNeS1kmNJU5ZYS8HVRhvu0TXuAY/UVC
ynEPUegkp+qYnR user@myemail.edu
```

16.2 Add or Replace Passphrase

In case you need to change your change passphrase, you can simply run `ssh-keygen -p` command. Then specify the location of your current key, and input (old and) new passphrases. There is no need to re-generate keys:

```
ssh-keygen -p
```

You will see the following output once you have completed that step:

```
Enter file in which the key is (/home/localname/.ssh/id_rsa):
Enter old passphrase:
Key has comment '/home/localname/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

16.3 SSH Add and Agent

To not always type in your password, you can use `ssh-add`.

It prompts the user for a private key passphrase and add it to a list of keys managed by the `ssh-agent`. Once it is in this list, you will not be asked for the passphrase as long as the agent is running with your public key.

To start the agent please use the following command:

```
eval `ssh-agent`
```

It is important that you use the backquote (`), located under the tilde (~), rather than the single quote ('). Once the agent is started it will print a PID that you can use to interact with later

To add the key use the command

```
ssh-add
```

To remove the agent use the command

```
kill $SSH_AGENT_PID
```

To execute the command upon logout, place it in your `.bash_logout` (assuming you use `bash`).

16.4 SSH Config

TODO: describe ssh config

16.5 SSH Port Forwarding

TODO: describe port forwarding

16.6 Upload the key to gitlab

Indiana University

We do not use `gitlab.com` for most classes but use `github.com`

In case you use gitlab Follow the instructions provided here to upload the public key

- <http://docs.gitlab.com/ce/ssh/README.html>

16.7 Using SSH on Mac OS X

Mac OS X comes with an ssh client. In order to use it you need to open the Terminal.app application. Go to Finder, then click Go in the menu bar at the top of the screen. Now click Utilities and then open the Terminal application.

16.8 Using SSH on Linux

All Linux versions come with ssh and can be used right from the terminal.

16.9 Using SSH on Raspberry Pi 3

Install Rasbian on an SD card, and boot up your system. Before putting it on the network, reset the default user password with the passwd command. Now you can use the terminal and use ssh just like on any Linux computer.

16.10 SSH on Windows

Warning

For this class we recommend that you use a virtual machine via virtual box and use the Linux ssh instructions. The information here is just provided for completeness and no support will be offered for native windows support.

Windows users need to have some special software to be able to use the SSH commands. If you have one that you are comfortable with and know how to setup key pairs and access the contents of your public key, please feel free to use it.

On Windows you have a couple of options on running Linux commands such as ssh. At this time it may be worth while to try the OpenSSH Client available for Windows, although it is in beta. If you like to use other methods we have included alternatives.

16.10.1 OpenSSH Client (Beta)

TODO: provide us with screenshots of the windows.

In case you need access to ssh Microsoft has fortunately updated their software to be able to run it directly from the Windows commandline including PowerShell.

However it is as far as we know not activated by default so you need to follow some setup scripts. Also this software is considered beta and its development and issues can be found at

- <https://github.com/PowerShell/Win32-OpenSSH>
- <https://github.com/PowerShell/Win32-OpenSSH/issues>

What you have to do is to install it by going to Settings > Apps and click Manage optional features under Apps & features.

Next, Click on the Add feature. You will be presented with a list in which you scroll down, till you find OpenSSH Client (Beta). Click on it and invoke Install.

After the install has completed, you can use the `ssh` command. Just type it in the commandshell or PowerShell

```
PS C:\Users\gregor> ssh
```

Naturally you can now use it just as on Linux or OSX. and use it to login to other resources

```
PS C:\Users\gregor> ssh myname@computer.example.com
```

16.10.2 Using SSH from Cygwin

One established way of using `ssh` is from using `cygwin`.

- <http://cygwin.com/install.html>

`Cygwin` contains a collection of GNU and Open Source tools providing Linux like functionality on Windows. A DLL is available that exposes the POSIX API functionality.

A list of supported commands is available at

- https://cygwin.com/packages/package_list.html

Please be minded that in order for `cygwin` to function easily the Windows user name should not include spaces. However, as the setup in windows encourages to use the full name whne you buy and setup a machine it may not be convenient to use. However, we just recommend that you create yourself a new username and use this if you like to use `cygwin`.

You can selectively install from the `cygwin` setup terminal which software you like to use, obviously you may want to use `ssh`

16.10.3 SSH from putty

As you will see the process is somewhat cumbersome and when you compare it with the command-line tools available, we do recommend using them instead.

`PuTTY` allows you to access the SSH, Telnet and Rlogin network protocols from windows.

- <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Although `PuTTY` has been out there for many years and served the community well, it is not following the standard `ssh` command line syntax when invoked from a command shell.

```
putty -ssh user@host.name
```

In addition to using `ssh`, it also provides a `copy` command.

```
pscp user@host.name:"\"remote filename with spaces\"" local_filename
```

`Putty` is best known for its GUI configuration application to manage several machines as demonstrated next. Once you have downloaded it and opened `PuTTYgen`, you will be presented with a key generator window (images provided by chameleon cloud) (see Figure 16.1).

To generate a key you click the *Generate* button which is blue. The `PuTTY` Key Generator (see Figure 16.2) will then ask you to move your mouse around the program's blank space to generate "randomness" for your key. You must enter a "Key passphrase" and then confirm the passphrase.

Next you need to save both the public and private keys into a file of your choice using the "Save public key" and "Save private key" buttons. We suggest you name something obvious like "public_key.pub" and "private_key" so that you can distinguish between the two.

Before closing this window, select the entire public key and copy it with "Control-C". Please note

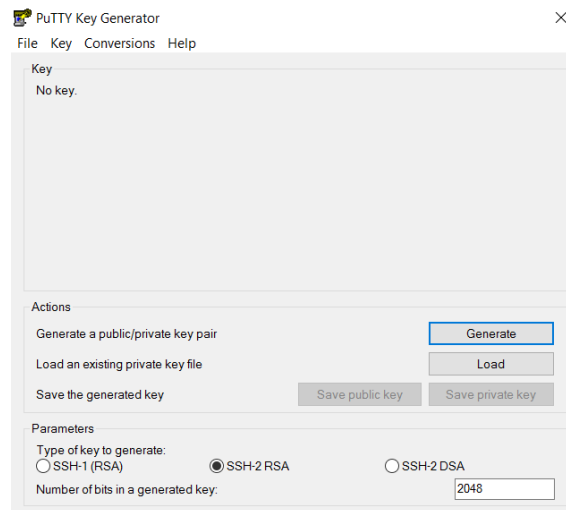


Figure 16.1: Key generation window

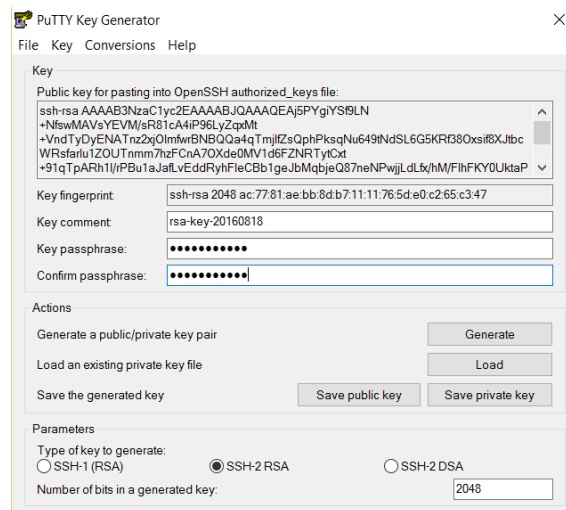


Figure 16.2: Key generation window

that everything should be copied, including “ssh-rsa”. This will be used when importing the key pair to Openstack.

At this time, the public key has been created and copied. Now you can use the public key and upload it to systems you like to login to.

16.10.4 Chocolatey

Another approach is to use it in Powershell with the help of chocolatey. Other options may be better suited for you and we leave it up to you to make this decision.

Chocolatey is a software management tool that mimics the install experience that you have on Linux and OSX. It has a repository with many packages. The packages are maintained by the community and you need to evaluate security implications when installing packages hosted on chocolatey just as you have to do if you install software on Linux and OSX from their repositories. Please be aware that there could be malicious code offered in the chocolatey repository although

the distributors try to remove them.

The installation is sufficiently explained at

- <https://chocolatey.org/install>

Once installed you have a command `choco` and you should make sure you have the newest version with:

```
choco upgrade chocolatey
```

Now you can browse packages at

- <https://chocolatey.org/packages>

Search for `openssh` and see the results. You may find different versions. Select the one that most suits you and satisfies your security requirements as well as your architecture. Lets assume you chose the Microsoft port, than you can install it with:

```
choco install openssh
```

Naturally, you can also install `cygwin` and `ptty` over `chocolatey`. A list of packages can be found at

- <https://chocolatey.org/packages>

Packages of interest include

- `emacs`: `choco install emacs`
 - <https://chocolatey.org/packages/Emacs>
- `pandoc`: `choco install pandoc`
 - <https://chocolatey.org/packages/pandoc>
- `LaTeX`: `choco install miktex`
 - <https://chocolatey.org/packages/miktex>
- `jabref`: `choco install jabref`
 - <https://chocolatey.org/packages/JabRef>
- `pycharm`: `choco install pycharm-community`
 - <https://chocolatey.org/packages/PyCharm-community>
- `lyx`: `choco install lyx`
 - <https://chocolatey.org/packages/lyx>
- `python 2`: `choco install python2`
 - <https://chocolatey.org/packages/python2>
- `python 3`: `choco install python`
 - <https://chocolatey.org/packages/python/3.6.4>
- `pip`: `choco install pip`
 - <https://chocolatey.org/packages/pip>
- `virtualbox`: `choco install virtualbox`
 - <https://chocolatey.org/packages/virtualbox>
- `vagrant`: `choco install vagrant`
 - <https://chocolatey.org/packages/vagrant>

Before installing any of them evaluate if you need them and identify security risks.

16.11 Tips

Use SSH keys You will need to use ssh keys to access remote machines

No blank passphrases In most cases you must use a passphrase with your key. In fact if we find that you use passwordless keys to futuresystems and to chameleon cloud resources, we may elect to give you an *F* for the assignment in question. There are some exceptions, but they will be clearly communicated to you in class. You will as part of your cloud drivers license test explain how you gain access to futuresystemes and chameleon to explicitly explain this point and provide us with reasons what you can not do.

A key for each server Under no circumstances copy the same private key on multiple servers. This violates security best practices. Create for each server a new private key and use their public keys to gain access to the appropriate server.

Use SSH agent So as to not to type in all the time the passphrase for a key, we recommend using ssh-agent to manage the login. This will be part of your cloud drivers license.

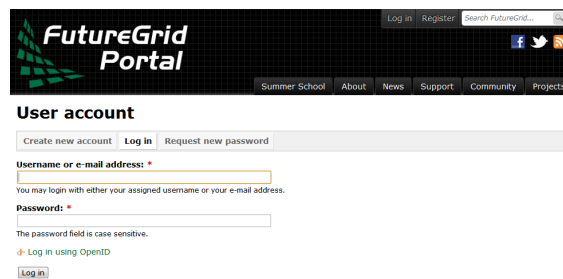
But shut down the ssh-agent if not in use

keep an offline backup, put encrypt the drive You may for some of our projects need to make backups of private keys on other servers you set up. If you like to make a backup you can do so on a USB stick, but make sure that access to the stick is encrypted. Do not stor anything else on that key and look it in a safe place. If you lose the stick, recreate all keys on all machines.

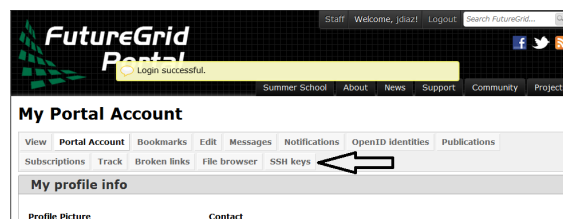
16.12 SSH to FutureSystems Resources

Next, you need to upload the key to the portal. You must be logged into the portal to do so.

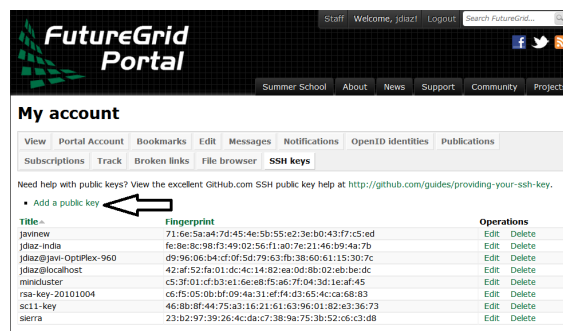
Step 1: Log into the portal



Step 2: Click in the “ssh key” button or go directly to <https://portal.futuresystems.org/my/ssh-keys>



Step 3: Click in the “add a public key” link.



Step 4: Paste your ssh key into the box marked Key. Use a text editor to open the “id_rsa.pub”. Copy the entire contents of this file into the ssh key field as part of your profile information. Many errors are introduced by users in this step as they do not paste and copy correctly.

FutureGrid Portal

Staff Welcome, juliaz Logout Search FutureGrid

Summer School About News Support Community Projects

Add a SSH key

Need help with public keys? View the excellent GitHub.com SSH public key help at <http://github.com/guides/providing-your-ssh-key>.

Title:

If this field is left blank, the key's title will be automatically generated.

Key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCK3H2G2FMhKc6T/U7UB8k16KRH4kUjgw9sc4Uu+Uwe/EwDwK6CBQMB+HKD9upCRW/851UyRjagt
R2eCRmZPC0VvH7VZhpJpT0W11088kL197VIVBP5AN3wVYNAJYjB/AUJW4GKXtmfKp98T30W4muatT1H4JAMGT+QzP7cama/fwaUeDAV
KZY7Agp7EBE6xwep7R8H7p7W03W05T0u68V7EakJFRmVfKChqP74WREGLU7WwemaA+15H48qQ36Fccp950Z7+H9d8es4emKucZY
SBHVRhuoTXuAY/UvcyEPLuegpg+qYR: JavibJav-PC
```

Step 5: Click the submit button. **IMPORTANT:** Leave the Title field blank. Make sure that when you paste your key, it does not contain newlines or carriage returns that may have been introduced by incorrect pasting and copying. If so, please remove them.

At this point, you have uploaded your key. However, you will still need to wait till all accounts have been set up to use the key, or if you did not have an account till it has been created by an administrator. Please, check your email for further updates. You can also refresh this page and see if the boxes in your account status information are all green. Then you can continue.

16.13 Testing your ssh key

If you have had no FutureSystem account before, you need to wait for up to two business days so we can verify your identity and create the account. So please wait. Otherwise, testing your new key is almost instantaneous on india. For other clusters like it can take around 30 minutes to update the ssh keys.

To log into india simply type the usual ssh command such as:

```
$ ssh portalname@india.futuresystems.org
```

The first time you ssh into a machine you will see a message like this:

```
The authenticity of host 'india.futuresystems.org (192.165.148.5)' can't be established.
RSA key fingerprint is 11:96:de:b7:21:eb:64:92:ab:de:e0:79:f3:fb:86:dd.
Are you sure you want to continue connecting (yes/no)? yes
```

You have to type yes and press enter. Then you will be logging into india. Other FutureSystem machines can be reached in the same fashion. Just replace the name india, with the appropriate FutureSystems resource name.

16.14 References

- [The Secure Shell: The Definitive Guide, 2 Ed \(O'Reilly and Associates\)](#)
- [putty](#)

16.15 Exercises

Exercise 16.1 create an SSH key pair

Exercise 16.2 upload the public key to git repository you use. Create a fork in git and use your ssh key to clone and commit to it

Exercise 16.3 Get an account on `futuresystems.org` (if you are authorized to do so). Upload your key to `futuresystems.org`. Login to `india.futuresystems.org`. Note that this could take some time as administrators need to approve you. Be patient. ■



17. Github

In some classes the material may be openly shared in code repositories. This includes class material, papers and project. Hence, we need some mechanism to share content with a large number of students. For this reason we use

- github.com

First, we like to introduce you to git and github.com (Section 17.1). Next, we provide you with the basic commands to interact with git from the commandline (Section 17.12). Then we will introduce you how you can contribute to this set of documentations with pull requests.

17.1 Overview

Github is a code repository that allows the development of code and documents with many contributors in a distributed fashion. There are many good tutorials about github. Some of them can be found on the github Web page. An interactive tutorial is for example available at

- <https://try.github.io/>

However, although these tutorials are helpful in many cases they do not address some cases. For example, you have already a repository set up by your organization and you do not have to completely initialize it. Thus do not just replicate the commands in the tutorial, or the once we present here before not evaluating their consequences. In general make sure you verify if the command does what you expect **before** you execute it.

A more extensive list of tutorials can be found at

- <https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github>

The github foundation has a number of excellent videos about git. If you are unfamiliar with git and you like to watch videos in addition to reading the documentation we recommend these videos

- <https://www.youtube.com/user/GitHubGuides/videos>

Next, we introduce some important concepts used in github.

17.2 Upload Key

Before you can work with a repository in an easy fashion you need to upload a public key in order to access your repository. Naturally, you need to generate a key first which is explained in


TODO: lessons-ssh-generate-key

before you upload one. Copy the contents of your `.ssh/id_rsa.pub` file and add them to [your github keys](#).

More information on this topic can be found on the [github Web page](#).

17.3 Fork


Forking is the first step to contributing to projects on GitHub. Forking allows you to copy a repository and work on it under your own account. Next, creating a branch, making some changes, and offering a pull request to the original repository, rounds out your contribution to the open source project.

Fork (1:41) 

17.4 Rebase

When you start editing your project, you diverge from the original version. During your developing, the original version may be updated, or other developers may have some of their branches implementing good features that you would like to include in your current work. That is when *Rebase* becomes useful. When you *Rebase* to certain points, could be a newer Master or other custom branch, consider you graft all your on-going work right to that point.

Rebase may fail, because some times it is impossible to achieve what we just described as conflicts may exist. For example, you and the to-be-rebased copy both edited some common text section. Once this happens, human intervention needs to take place to resolve the conflict.

Rebase (4:20) 

17.5 Remote

Collaborating with others involves managing the remote repositories and pushing and pulling data to and from them when you need to share work. Managing remote repositories includes knowing how to add remote repositories, remove remotes that are no longer valid, manage various remote branches and define them as being tracked or not, and more.

Though out this semester, you will typically work on two *remote* repos. One is the office class repo, and another is the repo you forked from the class repo. The class repo is used as the centralized, authority and final version of all student submissions. The repo under your own Github account is for your personal storage. To show progress on a weekly basis you need to commit your changes

on a weekly basis. However make sure that things in the master branch are working. If not, just use another branch to conduct your changes and merge at a later time. We like you to call your development branch `dev`.

- <https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes>


17.6 Pull Request

Pull requests are a means of starting a conversation about a proposed change back into a project. We will be taking a look at the strength of conversation, integration options for fuller information about a change, and cleanup strategy for when a pull request is finished.

Pull Request (4:26) 

17.7 Branch

Branches are an excellent way to not only work safely on features or experiments, but they are also the key element in creating Pull Requests on GitHub. Lets take a look at why we want branches, how to create and delete branches, and how to switch branches in this episode.

Branch (2:25) 


17.8 Checkout

Change where and what you are working on with the checkout command. Whether we are switching branches, wanting to look at the working tree at a specific commit in history, or discarding edits we want to throw away, all of these can be done with the checkout command.

Checkout (3:11) 


17.9 Merge

Once you know branches, merging that work into master is the natural next step. Find out how to merge branches, identify and clean up merge conflicts or avoid conflicts until a later date. Lastly, we will look at combining the merged feature branch into a single commit and cleaning up your feature branch after merges.

Merge (3:11) 

17.10 GUI

Using Graphical User Interfaces can supplement your use of the command line to get the best of both worlds. GitHub for Windows and GitHub for Mac allow for switching to command line, ease of grabbing repositories from GitHub, and participating in a particular pull request. We will also see the auto-updating functionality helps us stay up to date with stable versions of Git on the command line.

GUI (3:47) 

There are many other git GUI tools available that directly integrate into your operating system finders, windows, . . . , or PyCharm. It is up to you to identify such tools and see if they are useful for you. Most of the people we work with use git from the command line, even if they use PyCharm, eclipse, or other tools that have built in git support. You can identify a tool that works best for you.

17.11 Windows

This is a quick tour of GitHub for Windows. It offers GitHub newcomers a brief overview of what this feature-loaded version control tool and an equally powerful web application can do for developers, designers, and managers using Windows in both the open source and commercial software worlds. More: <http://windows.github.com>

Windows (1:25) 

17.12 Git from the Commandline

Although github.com provides a powerful GUI and other GUI tools are available to interface with github.com, the use of git from the commandline can often be faster and in many cases may be simpler.

Git commandline tools can be easily installed on a variety of operating systems including Linux, OSX, and Windows. Many great tutorials exist that will allow you to complete this task easily. We found the following two tutorials sufficient to get the task accomplished:

- <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
- <https://www.atlassian.com/git/tutorials/install-git>

Although the later is provided by an alternate repository to github. The installation instructions are very nice and are not impacted by it. Once you have installed git you need to configure it.

17.13 Configuration

Once you installed Git, you can need to configure it properly. This includes setting up your username, email address, line endings, and color, along with the settings' associated configuration scopes.

Configuration (2:47) 

It is important that make sure that use the `git config` command to initialize git for the first time on each new computer system or virtual machine you use. This will ensure that you use on all resources the same name and e-mail so that git history and log will show consistently your checkins across all devices and computers you use. If you do not do this, your checkins in git do not show up in a consistent fashion as a single user. Thus on each computer execute the following commands:

```
$ git config --global user.name "Albert Zweistein"
$ git config --global user.email albert.zweistein@gmail.com
```

where you replace the information with the information related to you. You can set the editor to emacs with:

```
$ git config --global core.editor emacs
```

Naturally if you happen to want to use other editors you can configure them by specifying the command that starts them up. You will also need to decide if you want to push branches individually or all branches at the same time. It will be up to you to make what will work for you best. We found that the following seems to work best:

```
git config --global push.default matching
```

More information about a first time setup is documented at:

```
* http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup
```

To check your setup you can say:

```
$ git config --list
```

One problem we observed is that students often simply copy and paste instructions, but do not read carefully the error that is reported back and do not fix it. Overlooking the proper set of the push.default is often overlooked. Thus we remind you: **Please read the information on the screen when you set up.**

17.14 Upload your public key

Please upload your public key to the repository as documented in github, while going to your account and find it in settings. There you will find a panel SSH key that you can click on which brings you to the window allowing you to add a new key. If you have difficulties with this find a video from the github foundation that explains this.

17.15 Working with a directory that was provided for you

In case your course provided you with a github directory, starting and working in it is going to be real simple. If you are the only student working on this you still need to make sure that papers or programs you manage in the repository work and do not interfere with scripts that instructors may use to check your assignments. Thus it is good to still create a branch, work in the branch and then merge the branch into the master once you verified things work. After you merged you can push the content to the github repository.

Tip: Please use only **lowercase** characters in the directory names and no special characters such as @ ; / _ and spaces. In general we recommend that you avoid using directory names with capital letters spaces and _ in them. This will simplify your documentation efforts and make the URLs from git more readable. Also while on some OS's the directories *MyDirectory* is different from *mydirectory* on OSX it is considered the same and thus renaming from capital to lower case can not be done without first renaming it to another directory.

Your homework for submission should be organized according to folders in your clone repository. To submit a particular assignment, you must first add it using:

```
git add <name of the file you are adding>
```

Afterwards, commit it using:

```
git commit -m "message describing your submission"
```

Then push it to your remote repository using:

```
git push
```

If you want to modify your submission, you only need to:

```
git commit -m "message relating to updated file"
```

afterwards:

```
git push
```

If you lose any documents locally, you can retrieve them from your remote repository using:

```
git pull
```

17.16 README.yml and notebook.md

In case you take classes e516 and e616 with us you will have to create a README.yml and notebook.md file in the top most directory of your repository. It serves the purpose of identifying your submission for homework and information about yourself.

It is important to follow the format precisely. As it is yaml it is an easy homework to write a 4 line python script that validates if the README.yml file is valid. In addition you can use programs such as yamllint which is documented at

- <https://yamllint.readthedocs.io/en/latest/>

This file is used to integrate your assignments into a proceedings. An example is provided at

- <https://github.com/bigdata-i523/sample-hid000/blob/master/README.yml>

Any derivation from this format will not allow us to see your homework as our automated scripts will use the README.yml to detect them. Make sure the file does not contain any TABs. Please also mind that all filenames of all homework and the main directory must be **lowercase** and do not include spaces. This will simplify your task of managing the files across different operating systems.

In case you work in a team, on a submission, the document will only be submitted in the author and hid that is listed first. All other readmes, will have for that particular artifact a `duplicate: yes` entry to indicate that this submission is managed elsewhere. The team will be responsible to manage their own pull requests, but if the team desires we can grant access for all members to a repository by a user. Please be aware that you must make sure you coordinate with your team.

We will not accept submission of homework as pdf documents or tar files. All assignments must be submitted as code and the reports in native latex and in github. We have a script that will automatically create the PDF and include it in a proceedings. There is no exception from this rule and all reports not compilable will be returned without review and if not submitted within the deadline receive a penalty.

Please check with your instructor on the format of the README.yml file as it could be different for your class.

To see an example for the notebook.md file, you can visit our sample hid, and browse to the notebook.md file. Alternatively you can visit the following link

- <https://github.com/bigdata-i523/sample-hid000/blob/master/notebook.md>

The purpose of the notebook md file is to record what you did in the class to us. We will use this file at the end of the class to make sure you have recorded on a weekly basis what you did for the class. Inactivity is a valid response. Not updating the notebook, is not.

The sample directory contains other useful directories and samples, that you may want to investigate in more detail. One of the most important samples is the github issues (see Section 17.19). There is even a video in that section about this and showcases you how to organize your tasks within this

class, while copying the assignments from piazza into one or more github issues. As we are about cloud computing, using the services offered by a prominent cloud computing service such as github is part of the learning experience of this course.

17.17 Contributing to the Document

TODO: This section has to be redone as we use class specific clones and not the master

17.17.1 Clone

```
$ git remote add upstream https://github.com/cloudmesh/book
```

17.17.2 Merge

As we are allowing contribution by the community, they are best managed through a merge with our upstream repository so you can update to the newest status before you issue a pull request.

Make sure you have upstream repo defined:

```
$ git remote add upstream https://github.com/cloudmesh/book
```

Now Get latest from upstream:

```
$ git rebase upstream/master
```

In this step, the conflicting file shows up (in my case it was refs.bib):

```
$ git status
```

should show the name of the conflicting file:

```
$ git diff <file name>
```

should show the actual differences. May be in some cases, It is easy to simply take latest version from upstream and reapply your changes.

So you can decide to checkout one version earlier of the specific file. At this stage, the re-base should be complete. So, you need to commit and push the changes to your fork:

```
$ git commit
$ git rebase origin/master
$ git push
```

Then reapply your changes to refs.bib - simply use the backedup version and use the editor to redo the changes.

At this stage, only refs.bib is changed:

```
$ git status
```

should show the changes only in refs.bib. Commit this change using:

```
$ git commit -a -m "new:usr: <message>"
```

And finally push the last committed change:

```
$ git push
```

The changes in the file to resolve merge conflict automatically goes to the original pull request and the pull request can be merged automatically.

You still have to issue the pull request from the Github Web page so it is registered with the upstream repository.

17.17.3 Resources

- [Pro Git book](#)
- [Official tutorial](#)
- [Official documentation](#)
- [TutorialsPoint on git](#)
- [Try git online](#)
- [GitHub resources for learning git](#) Note: this is for github and not for gitlab. However as it is for gt the only thing you have to do is replace hihub, for gitlab.
- [Atlassian tutorials for git](#)

In addition the tutorials from atlasian are a good source. However remember that you may not use bitbucket as the repository, so ignore those tutorials. We found the following useful

- What is git: <https://www.atlassian.com/git/tutorials/what-is-git>
- Installing git: <https://www.atlassian.com/git/tutorials/install-git>
- git config: <https://www.atlassian.com/git/tutorials/setting-up-a-repository#git-config>
- git clone: <https://www.atlassian.com/git/tutorials/setting-up-a-repository#git-clone>
- saving changes: <https://www.atlassian.com/git/tutorials/saving-changes>
- collaborating with git: <https://www.atlassian.com/git/tutorials/syncing>

17.18 Exercises

Exercise 17.1 How do you set your favorite editor as a default with github config ■


Exercise 17.2 What is the difference between merge and rebase? ■

Exercise 17.3 Assume you have made a change in your local fork, however other users have since committed to the master branch, how can you make sure your commit works off from the latest information in the master branch? ■

Exercise 17.4 Find a spelling error in the Web page or a contribution and create a pull request for it. ■

Exercise 17.5 Create a README.yml in your github account directory provided for you for class. ■

17.19 Github Issues

Issues (8:29) 

When we work in teams or even if we work by ourselves, it is prudent to identify a system to coordinate your work. While conduction projects that use a variety of cloud services, it is important to have a system that enables us to have a cloud service that enables us to facilitate this coordination.

section/git/github.tex

Github provides such a feature through its *issue* service that is embedded in each repository.

Issues allow for the coordination of tasks, enhancements, bugs, as well as self defined labeled activities. Issues are shared within your team that has access to your repository. Furthermore, in an open source project the issues are visible to the community, allowing to easily communicate the status, as well as a roadmap to new features.

This enables the community to participate also in reporting of bugs. Using such a system transforms the development of software from the traditional closed shop development to a truly open source development encouraging contributions from others. Furthermore it is also used as bug tracker in which not only you, but the community can communicate bugs to the project.

TODO: Tyler: Include image of the issues on hid-sample

A good resource for learning more about issues is provided at

- <https://guides.github.com/features/issues/>

17.19.1 Git Issue Features

A git issue has the following features:

title -- a short description of what the issue is about

description a more detailed description. Descriptions allow also to conveniently add check-boxed todo's.

label a color enhanced label that can be used to easily categorize the issue. YOU can define your own labels.

milestone a milestone so you can identify categorical groups issues as well as their due date. You can for example group all tasks for a week in a milestone, or you could for example put all tasks for a topic such as developing a paper in a milestone and provide a deadline for it.

assignee an assignee is the person that is responsible for making sure the task is executed or on track if a team works on it. Often projects allow only one assignee, but in certain cases it is useful to assign a group, and the group identifies if the task can be split up and assigns them through check-boxed todo's.

comments allow anyone with access to provide feedback via comments.

17.19.2 Github Markdown

Github uses markdown which we introduce you in Section 14.2.

As github has its own flavor of markdown we however also point you to

- <https://help.github.com/articles/basic-writing-and-formatting-syntax/>

as a reference. We like to mention the special enhancements for github's markdown that integrate well to support project management.

Task lists

Task lists can be added to any description or comment in github issues. To create a task list you can add to any item []. This includes a task to be done. To make it as complete simply change it to [x]. Whoever the great feature of tasks is that you do not even have to open the editor but you can simply check the task on and off via a mouse click. An example of a task list could be

```
1 Post Bios
2
```

```

3 * [x] Post bio on piazza
4 * [ ] Post bio on google docs
5 * [ ] Post bio on github
6 * [ ] \optional) integrate image in google docs bio

```

In case you need to use a (have at the beginning of the task text, you need to escape it with a \

Team integration

A person or team on GitHub can be mentioned by typing the username proceeded by the @ sign. When posting the text in the issue, it will trigger a notification to them and allow them to react to it. It is even possible to notify entire teams, which are described in more detail at

- <https://help.github.com/articles/about-teams/>

Referencing Issues and Pull requests

Each issue has a number. If you use the # followed by the issue number you can refer to it in the text which will also automatically include a hyperlink to the task. The same is valid for pull requests.

Emojis

Although github supports emojis such as `:+1:` we do not use them typically in our class.

17.19.3 Notifications

GitHub allows you to set preferences on how you like to receive notifications. You can receive them either via e-mail or the Web. This is controlled by configuring it in *your settings*, where you can set the preferences for participating projects as well as projects you decide to watch. To access the notifications you can simply look at them in the *notification* screen. In this screen when you press the ? you will see a number of commands that allow you to control the notification when pressing on one of them.

17.19.4 cc

To carbon copy users in your issue text, simply use `/cc` followed by the @ sign and their github user name.

17.19.5 Interacting with issues

GitHub has the ability to search issues with a search query and a search language that you can find out more about it at

<https://guides.github.com/features/issues/#search>

A dashboard gives convenient overviews of the issues including a *pulse* that lists todo's status if you use them in the issue description.



18. Virtual Box

For development purposes we recommend that you use for this class an Ubuntu virtual machine that you set up with the help of virtualbox. We recommend that you use the current version of ubuntu and do not install or reuse a version that you have set up years ago.

As access to cloud resources requires some basic knowledge of linux and security we will restrict access to our cloud services to those that have demonstrated responsible use on their own computers. Naturally as it is your own computer you must make sure you follow proper security. We have seen in the past students carelessly working with virtual machines and introducing security vulnerabilities on our clouds just because “it was not their computer.” Hence, we will allow using of cloud resources only if you have demonstrated that you responsibly use a linux virtual machine on your own computer. Only after you have successfully used ubuntu in a virtual machine you will be allowed to use virtual machines on clouds.

A “cloud drivers license test” will be conducted. Only after you pass it we will let you gain access to the cloud infrastructure. We will announce this test. Before you have not passed the test, you will not be able to use the clouds. Furthermore, you do not have to ask us for join requests to cloud projects before you have not passed the test. Please be patient. Only students enrolled in the class can get access to the cloud.

If you however have access to other clouds yourself you are welcome to use them, However, be reminded that projects need to be reproducible, on our cloud. This will require you to make sure a TA can replicate it.

Let us now focus on using virtual box.

18.1 Installation

First you will need to install virtualbox. It is easy to install and details can be found at

<section/virtualbox.tex>

- <https://www.virtualbox.org/wiki/Downloads>

After you have installed virtualbox you also need to use an image. For this class we will be using ubuntu Desktop 16.04 which you can find at:

- <http://www.ubuntu.com/download/desktop>

Please note some hardware you may have may be too old or has too little resources to be useful. We have heard from students that the following is a minimal setup for the desktop machine:

- multi core processor or better allowing to run hypervisors
- 8 GB system memory
- 50 GB of free hard drive space

For virtual machines you may need multiple, while the minimal configuration may not work for all cases.

As configuration we often use

minimal 1 core, 2GB Memory, 5 GB disk

latex 2 core, 4GB Memory, 25 GB disk

A video to showcase such an install is available at:

[Video \(seconds\)](#) 

Note

If you specify your machine too small you will not be able to install the development environment. Gregor used on his machine 8GB RAM and 25GB disk space.

Please let us know the smallest configuration that works.


18.2 Guest additions

The virtual guest additions allow you to easily do the following tasks:

- Resize the windows of the vm
- Copy and paste content between the Guest operating system and the host operating system windows.

This way you can use many native programs on you host and copy contents easily into for example a terminal or an editor that you run in the Vm.

A video is located at

[Video \(4:46\)](#) 

Please reboot the machine after installation and configuration.

On OSX you can once you have enabled bidirectional copying in the Device tab with

OSX to VBox: command c shift CONTRL v

Vbox to OSX: shift CONTRL v shift CONTRL v

Note

On Windows the key combination is naturally different. Please consult your windows manual. If you let us know TAs will add the information here.

18.3 Exercises

Exercise 18.1 Install ubuntu desktop on your computer with guest additions. ■

Exercise 18.2 Make sure you know how to paste and copy between your host and guest operating system. ■

Exercise 18.3 Install the programs defined by the development configuration. ■

Exercise 18.4 Provide us with the key combination to copy and paste between Windows and Vbox. ■



Cloud Resources

19	FutureSystems	241
19.1	FutureSystems evolved from FutureGrid	
19.2	Creating Portal Account	
19.3	SSH Key Generation using ssh-keygen command	
19.4	Shell Access via SSH	
19.5	Advanced SSH	
19.6	SSH Key Generation via putty	
19.7	FutureSystems Facilities	
20	Chameleon Cloud	247
20.1	Overview	
20.2	Resources	
20.3	Hardware	
20.4	Getting Started	
20.5	Charge Rates	
20.6	OpenStack Virtual Machines	
20.7	Horizon Graphical User Interface	
20.8	HEAT	
20.9	Bare Metal	
20.10	Frequently Asked Questions	



19. FutureSystems

This section gives an overview of the FutureSystems that are available as part of the DSC infrastructure. We cover the creation of FutureSystems Account, Uploading SSH Key and how to instantiate and log into Virtual Machine and accessing Ipython are covered. In the end we discuss about running Python and Java on Virtual Machine.

19.1 FutureSystems evolved from FutureGrid

In this video we introduce FutureGrid a precursor to FutureSystems.

[FutureGid \(12:12\)](#) 

At this time we are replacing several of the older systems. To use these new systems you need to ask for access through them via our portal.

19.2 Creating Portal Account

This lesson explains how to create a portal account, which is the first step in gaining access to FutureSystems.

See Lesson 4 and 7 for SSH key generation on Linux, OSX or Windows.

[FutureGrid Introduction \(11:50\)](#) 

19.3 SSH Key Generation using ssh-keygen command

SSH keys are used to identify user accounts in most systems including FutureSystems. This lesson walks you through generating an SSH key via ssh-keygen command line tool.

[section/futuresystems.tex](#)

[ssh-key gen \(4:06\)](#) 

19.4 Shell Access via SSH

This lesson explains how to get access FutureSystems resources vis SSH terminal with your registered SSH key.

[Shell Access via SSH \(2:34\)](#) 

19.5 Advanced SSH

This lesson shows you how to write SSH ‘config’ file in advanced settings.

[Advanced SSH \(2:47\)](#) 

19.6 SSH Key Generation via putty

This lesson is for Windows users. You will learn how to create an SSH key using PuTTYgen, add the public key to you FutureSystems portal, and then login using the PuTTY SSH client.

[Windows users \(3:51\)](#) 

19.7 FutureSystems Facilities

FutureSystems system resources are located at Indiana University (Bloomington). Resources at Indiana University (Bloomington) include a

1. 128-core HP cluster (Bravo),
2. 92-core cloud cluster (Echo),
3. 192-core Tesla GPU cluster (Delta),
4. 3456-core Haswell cluster (Juliet),
5. 126-core NVIDIA K80/Volta GPU cluster (Romeo),
6. 3264-core Knight’s Landing cluster (Tango),
7. 480-core Platinum cluster (Tempest),
8. 768-core Platinum cluster (Victor).

Details of the resources are listed in subsequent sections.

19.7.1 Bravo

The large-memory HP cluster (Bravo) is a 1.7 Tflop HP Proliant distributed shared memory cluster with 128 processor cores and 3 TB total memory capacity. The compute nodes consist of 16 HP DL180 servers, each with two quad-core Intel Xeon E5620 2.4 GHz processors, 192 GB of memory, 12 TB of local attached storage, and a PCIe 4x QDR InfiniBand adapter for high bandwidth, low-latency MPI applications. Br avo is currently used as a shared storage cluster and is not being utilized for compute jobs. Operating System: RedHat Linux 6.9.

[section/futuresystems.tex](#)

19.7.2 Delta

The GPU cluster (Delta) is a SuperMicro distributed shared memory cluster with 192 CPU cores and 14,336 GPU cores and 3TB total memory capacity. The compute nodes consist of 16 SuperMicro X8DTG-QF servers, each with 2 6-core Intel Xeon 5660 2.80 GHz processors, 2 nVIDIA Tesla C2075 GPUs with 448 cores per GPU, 192GB of memory, 9TB of local attached storage, and a Mellanox ConnectX-2 VPI dual-port InfiniBand QDR/10GigE PCIe adapter card. Operating System: RedHat Linux 7.4

19.7.3 Echo

The cloud cluster (Echo) is a SuperMicro distributed shared memory cluster with 192 CPU cores and 6TB total memory capacity. The compute nodes consist of 16 SuperMicro X9DRW servers, each node with 2 6-core Intel(R) Xeon(R) CPU E5-2640 2.50GHz processors; 384GB of memory, 10TB of local disk storage, a 10GbE Ethernet and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s onboard adapter for high bandwidth, low-latency MPI applications. Operating System: Ubuntu Linux 16.04

19.7.4 Juliet

The Haswell cluster (Juliet) is a SuperMicro distributed shared memory cluster with 3456 CPU cores and 16TB total memory capacity. The compute nodes consist of SuperMicro X10DRT-HIBF servers, 32 nodes with 2 18-core Intel(R) Xeon(R) CPU E5-2699 v3 2.30GHz processors; 96 nodes with 2 12-core Intel(R) Xeon(R) CPU E5-2670 v3 2.30GHz processors, all compute nodes with 128GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s onboard adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4

19.7.5 Romeo

The K80/Volta GPU cluster (Romeo) is a SuperMicro distributed shared memory cluster with 126 CPU cores, 161792 CUDA cores, and 768GB total memory capacity. The compute nodes consist of 4 SuperMicro X10DGQ servers with 2 12-core Intel(R) Xeon(R) CPU E5-2670 v3 2.30GHz processors, 4 NVIDIA GK210GL [Tesla K80] GPU Accelerator cards with 4992 CUDA cores, and 2 SuperMicro X10DGO servers with 2 10-core Intel Xeon E5-2600 v4 2.2GHz processors and 8 NVIDIA V100 (Tesla Volta) accelerators with 5120 CUDA cores. All nodes with 128GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s onboard adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4

19.7.6 Tango

The Knight's Landing cluster (Tango) is a Penguin Computing distributed shared memory cluster with 3264 Xeon Phi cores and 12.8TB total memory capacity. The compute nodes consist of 16 nodes with 1 72-core Intel(R) Xeon Phi(TM) CPU 7290F 1.50GHz processor and 48 nodes with 1 68-core Intel(R) Xeon Phi(TM) CPU 7250F 1.50GHz processor. All nodes with 200GB of memory, 3.2TB of local disk storage, 800GB of NVMe storage, and an Intel OmniPath adapter for high bandwidth, low-latency MPI applications. Operating System: CentOS release 7.2.1511

19.7.7 Tempest

The Platinum cluster (Tempest) is a SuperMicro distributed shared memory cluster with 480 CPU cores and 2.5TB total memory capacity. The 10 compute nodes consist of SuperMicro X11DPT-PS servers with 2 24-core Intel(R) Xeon(R) Platinum 8160 2.10GHz processors, 256GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and an Intel OmniPath adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4

19.7.8 Victor

The Platinum cluster (Victor) is a SuperMicro distributed shared memory cluster with 768 CPU cores and 2.5TB total memory capacity. The 16 compute nodes consist of SuperMicro X11DPT-PS servers with 2 24-core Intel(R) Xeon(R) Platinum 8160 2.10GHz processors, 256GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4 (see Figure 19.1).

19.7.9 PI Cluster

The details will be added once we have completed the purchase.

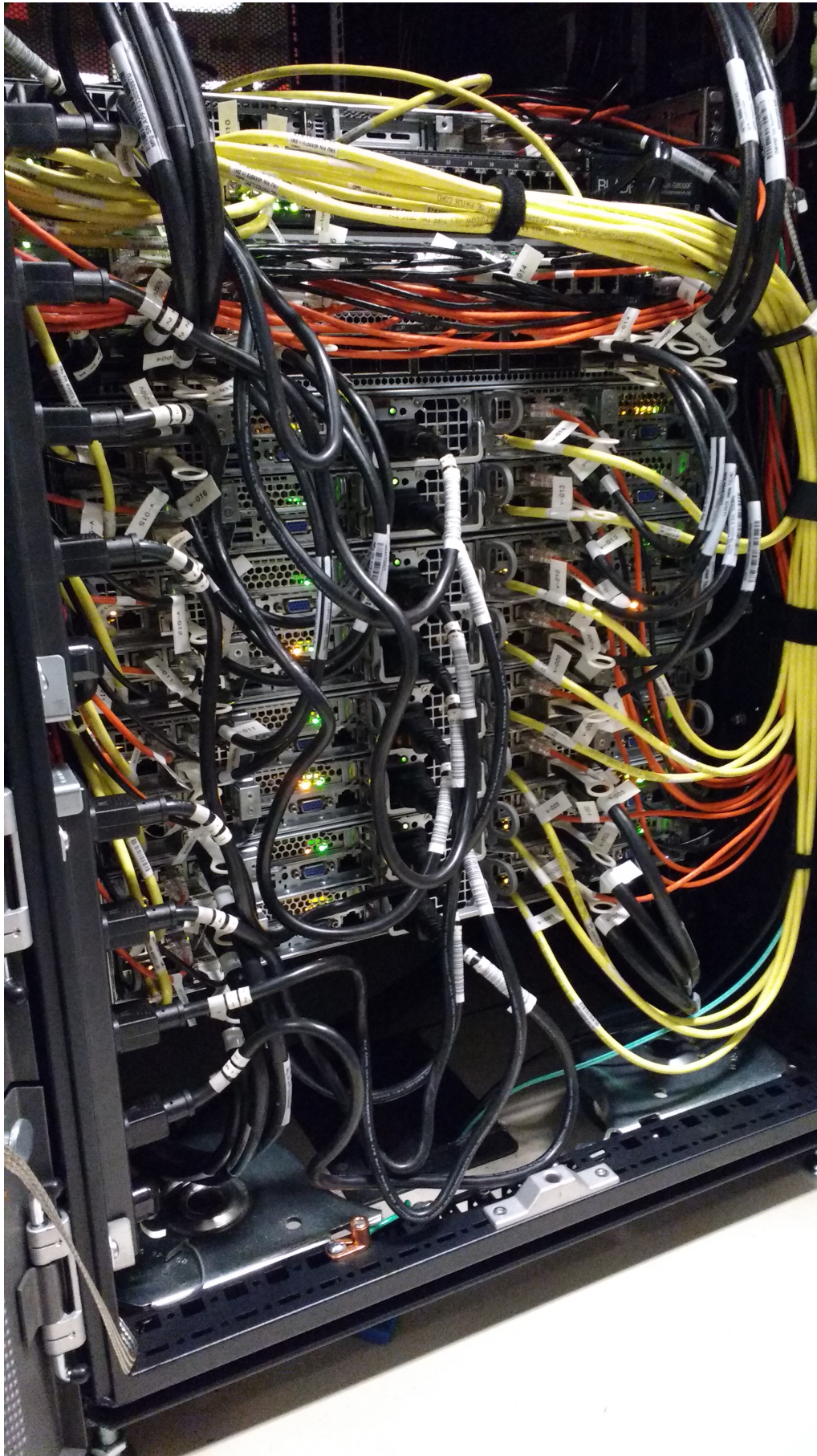


Figure 19.1: Cabeling of Victor



20. Chameleon Cloud

20.1 Overview

This chapter is copied from the Chameleon Web page. However we have included where appropriate some updates.

Chameleon is an experimental testbed for Computer Science funded by the NSF FutureCloud program. Chameleon is built over two sites, University of Chicago and TACC, offering a total of over 550 nodes and 5 PB of space in twelve [Standard Cloud Unit \(SCU\) racks](#). To effectively support Computer Science experiments Chameleon offers bare metal reconfigurability on most of the hardware. To provide easy access to educational users, three SCUs at TACC (a quarter of the testbed) are configured with OpenStack KVM. You can read more about Chameleon [here](#).

Chameleon is broadly available to members of the US Computer Science research community and its international collaborators working in the open community on cloud research. The expectation is that any research performed on Chameleon will result in publication in a broadly available journal or conference.

In order to promote fairness to all users, we have the following set of Best Practices for using Chameleon bare metal partitions:

Think Small for Development and class use: If you are just developing or prototyping a system, and not yet running experiments at scale, use only as many nodes as you actually need (e.g., many projects can be developed and tested on 3-4 nodes), and try to take short reservations. Start with hours first and do not let your VMs unnecessarily run for long unused periods.

Automate deployments: You can always snapshot your work/images between sessions using the [snapshotting instructions](#) to simplify the redeployment of your environment during the next

work session. You can also use scripting and environment customization to make it easier to redeploy images. An additional benefit of automation is that it makes it easier for you to reproduce your work and eventually share it with colleagues within your lab and other collaborators.

Think Big for Experimentation: Once you are ready to experiment you will want to test your experimental setup on increasingly larger scales. This is possible by taking an advance reservation for many resources for a relatively short time. The more resources you need, the more likely it is that you will need to run experiments at a less attractive time (e.g., during the weekend) --- here's where automation will also help. In justified cases, we will support reserving even the whole bare metal testbed.

20.2 Resources

In any case please visit the Chameleon Web page as there is also more information about other topics that we may not care about. Furthermore we do not use Advanced Appliances, but use ansible instead as it is independent from OpenStack and can be used with other frameworks.

If you prefer you can also go to the Chameleon Web site using the following links. However we have improved some of the documentation found in this document. We would like to get your feedback in case you find errors or like to contribute to this documentation.

Warning

Chameleon cloud promotes insecure use of ssh while suggesting passphrase less keys. This is **very dangerous** due to the fact that someone could gain access to your computer and if a password less key is stolen easy access to other systems can be achieved. Instead you must use whenever possible passphrases and use ssh agent and ssh add!

Hence do not use their advise that is mentioned multiple times in their documentation. Follow ours!

The links to Chameleons Documentation

- [Home](#)
- [Documentation](#)
 - [Getting Started](#)
 - [Bare Metal](#)
 - [Complex Appliances](#)
 - [OpenStack KVM Cloud](#)
 - [User FAQ](#)
 - [Community](#)
- [Appliances](#)
- [Hardware](#)
- [News](#)
- [About](#)
 - [About Chameleon](#)
 - [Hardware Description](#)
 - [Talks](#)
 - [Stay in Touch](#)
 - [Media Resources](#)
- [Log in](#)

- [Users](#)
 - [Register](#)
 - [Experiment](#)
 - [Help](#)

20.3 Hardware

The Chameleon architecture consists of a set of standard cloud units (SCUs), each of which is a single rack with 42 compute nodes, 4 storage nodes attached to 128TB of local storage in configurable arrays, and an OpenFlow compliant network switch. In addition to the homogeneous SCUs, a variety of heterogeneous hardware types is available to experiment with alternative technologies. The testbed also includes a shared infrastructure with a persistent storage system accessible across the testbed, a top-level network gateway to allow access to public networks, and a set of management and provisioning servers to support user access, control, monitoring and configuration of the testbed. Chameleon is physically distributed between the Texas Advanced Computing Center (TACC) and the University of Chicago (UC) through 100Gbps Internet2 links, to allow users to examine the effects of a distributed cloud.

Hardware Summary

Standard Cloud Units (SCUs)	Homogeneous Hardware Types
Number of Nodes per Rack:	42 Compute Nodes 4 Storage Nodes
Local Storage per homogeneous SCU:	128TB (configurable)
Network Switch:	OpenFlow Compliant
TACC/UC Distributed Cloud	100Gbps Internet2 links

[Detailed information in our Resource Discovery Portal](#)

20.3.1 Standard Cloud Units

The homogeneous standard cloud unit is a self-contained rack with all the components necessary to run a complete cloud infrastructure, and the capability to combine with other units to form a larger experiment. The rack consists of 42 Dell R630 servers; each with 24 cores delivered in dual socket Intel Xeon E5-2670 v3 ‘Haswell’ processors (each with 12 cores @ 2.3GHz) and 128 GiB of RAM. In addition to the compute servers, each unit contains storage hosted in two FX2 chassis, each containing two Dell FC430 servers attached to two Dell PowerEdge FD332 storage blocks containing 16 2TB hard drives, for a total of 128TB of raw disk storage per unit. These FC430 storage nodes contain dual socket Intel Xeon E5-2650 v3 ‘Haswell’ processors (each with 10 cores @ 2.3 GHz), 64 GiB of RAM, and can be combined across SCUs to create a Big Data infrastructure with more than a PB of storage. Each node in the SCU connects to a Dell switch at 10Gbps, with 40Gbps of bandwidth to the core network from each SCU. The total system contains 12 SCUs (10 at TACC and 2 at UC) for a total of 13,056 cores, 66 TiB of RAM, and 1.5PB of configurable storage in the SCU subsystem.

20.3.2 Network

Networking is changing rapidly, and the network fabric is as much a part of the research focus of Chameleon as the compute or storage. For the Chameleon network, every switch in the research

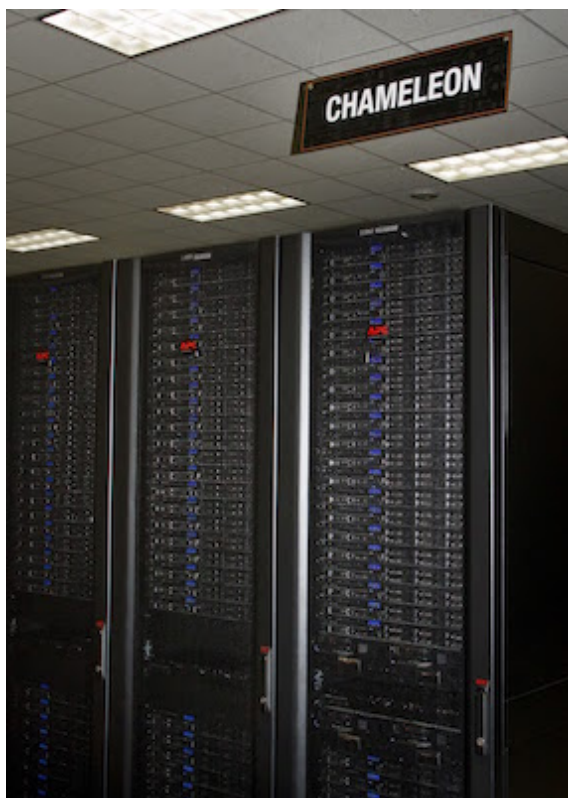


Figure 20.1: Chameleon Cloud Racks

network is a fully OpenFlow compliant programmable Dell S6000-ON switch. Each node connects to this network at 10 Gbps, and each unit uplinks with 40Gbps per rack to the Chameleon core network. The core switches (Dell S6000-ON) are connected by 40 Gbps Ethernet links, which connect to the backbone 100Gbps services at both UC and TACC. A Fourteen Data Rate (FDR) Infiniband network (56Gbps) is also deployed on one SCU to allow exploration of alternate networks.

20.3.3 Shared Storage

While storage is dynamically provisioned to researchers to be used as an experiment needs within the SCUs, Chameleon also provides a shared storage system. The shared storage provides more than 3.6PB of raw disk in the initial configuration, which is partitioned between a file system and an object store that is persistent between experiments. The shared storage is comprised of four Dell R630 servers with 128 GiB of RAM, four MD3260 external drive arrays, and six MD3060e drive expansion chassis, populated by 600 6TB near line SAS drives. The system also includes a dozen PowerEdge R630 servers as management nodes to provide for login access to the resource, data staging, system monitoring, and hosting various OpenStack services.

20.3.4 Heterogeneous Compute Hardware

The heterogeneous hardware includes various technologies: GPU and FPGA accelerators, SSD and NVMe storage, low-power ARM, Atom, and Xeon systems-on-a-chip. With the exception of the low-power systems-on-a-chip, each of the additional nodes is a Dell PowerEdge R730 server with the same CPUs as the R630 servers in our SCUs.

The two storage hierarchy nodes have been designed to enable experiments using multiple layers of caching: they are configured with 512 GiB of memory, two Intel P3700 NVMe of 2 TB each, four Intel S3610 SSDs of 1.6 TB each, and four 15K SAS HDDs of 600 GB each.

The GPU offering consists of two K80 GPU nodes, two M40 GPU nodes, sixteen P100 GPU nodes. These nodes target experiments using accelerators to improve the performance of some algorithms, experiments with new visualization systems, and deep machine learning. Each K80 GPU node is upgraded with an NVIDIA Tesla K80 accelerator, consisting of two GK210 chips with 2496 cores each (4992 cores in total) and 24 GiB of GDDR5 memory. Each M40 node is upgraded with an NVIDIA Tesla M40 accelerator, consisting of a GM200 chip with 3072 cores and 12 GiB of GDDR5 memory. The P100 nodes have two GPU cards installed each, providing 32 P100 GPUs in total. The P100 GPUs utilize GP100 chips providing 3584 cores, with 16 GiB GDDR5 RAM in each card. In order to make it easy for users to get started with the GPU nodes, we have developed a [CUDA appliance](#) that includes NVIDIA drivers as well as the CUDA framework.

GPU	Chip	Cores per GPU	RAM per GPU	GPU per node	# of nodes
Tesla K80	GK 210	2496 × 2	24 GiB GDDR5	1	2
Tesla M40	GM 200	3072	12 GiB GDDR5	1	2
Tesla P100	GP100	3584	16 GiB GDDR5	2	16

The four FPGA nodes have a Nallatech 385A board with an Altera Arria 10 1150 GX FPGA (up to 1.5 TFlops), 8 GiB DDR3 on-card memory, and dual QSFP 10/40 GbE support. The [Chameleon FPGA User Guide](#) provides details for conducting experiments on this hardware.

The low-power systems are comprised of 8 low power Xeon servers (HP ProLiant m710p with one 4-core Intel Xeon E3-1284L v4 processor), 8 Atom servers (HP ProLiant m300 with one 8-core Intel Avoton-based System on a Chip), and 24 ARM servers (HP ProLiant m400 with one 8-core AppliedMicro X-gene System on a Chip). These are all delivered in a single HP Moonshot 1500 chassis.

For more information on how you can reserve these nodes, see the [heterogeneous hardware section](#) of the bare metal user's guide.

20.3.5 Live updates

You can browse detailed information about the resources offered for bare metal reconfiguration in our [Resource Discovery Portal](#).

20.4 Getting Started

We describe how you can get access to chameleon cloud under the assumption that you are a student or a researcher that joins an existing project on Chameleon cloud. You will need to follow the following steps:

20.4.1 Step 1: Create a Chameleon account

To get started using Chameleon you will need to [create a user account](#).

You will be asked to agree to the [Chameleon terms and conditions](#) which, among others, ask you to acknowledge the use of Chameleon in your publications.

Acknowledgement of support from the Chameleon project and the National Science Foundation should appear in any publication of material, whether copyrighted or not, that describes work which benefited from access to Chameleon cyberinfrastructure resources. The suggested acknowledgement is as follows: “Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation”.

Indiana University

As part of creating an account you may request PI status. However you are not a PI as you will be joining a project.

20.4.2 Step 2: Create or join a project

To use Chameleon, you will need to be associated with a [project](#) that is assigned an [allocation](#). This means that you either need to

1. [apply for a new project](#) or
2. [ask the PI of an existing Chameleon project to add you.](#)

A project is headed by a project PI, typically a [faculty member or researcher scientist at a scientific institution](#). If you are a student we recommend that you ask your professor to work with you on creating a project. Please note that you must not create a project by yourself and that you indeed need to work with your professor.

In case you need to do a project application typically consists of about one paragraph description of the intended research and takes one business day to process.

Enrolling you into an existing research or class project depends on the time availability of the project lead or professor of your class. It is important that you communicate your chameleon cloud account name to the project lead so they can easily add you. Make sure you really give them only your chameleon account name and potentially your organizational e-mail, Firstname, and Lastname so they can check you are eligible to get access.

Indiana University

Indiana University students that take the e516 and e616 classes will have to fill out a google form in which they communicate the chameleon cloud name. You can already apply for an account name, but do not apply for a project. If you nevertheless apply for a project, we will hear from the administrators and you will receive a point deduction.

20.4.3 Step 3: Start using Chameleon!

Now that you have enrolled and once you are added to the project by your project lead you can start using chameleon cloud. However be reminded that you ought to shut down the resources/VMs whenever they are not in use to avoid unnecessary charging. Remember the project has limited time on chameleon and any unused time will be charged against the project.

Chameleon provides two types of resources with links to their respective users guides below:

[Bare Metal User Guide](#) will tell you how to use Chameleon bare metal resources which provide strong isolation and allow you maximum control (reboot to new operating system, reboot the kernel, etc.)

[OpenStack KVM User Guide](#) will tell you how to get started with Chameleon's OpenStack KVM cloud which is a multi-tenant environment providing weak performance isolation.

If you have any questions or encounter any problems, you can check out our [User FAQ](#), or [submit a ticket](#).

Indiana University

As part of the classes you will need to first pass a cloud *security* drivers licence test. The test is designed so that you think about gaining access to a VM securely and how to properly secure the VM. Once passed, access is typically provided by midterm time. You are not allowed to constantly run VM's and must shut them down if not in use. You will get point deductions if we detect you do not obey by this rule. We have access to log files about your VM usage.

20.5 Charge Rates

It is important to fully understand the charge rates of your VM and storage use.

Chameleon has two types of limitations, introduced to promote fair resource usage to all:

Allocation: Chameleon projects are limited to a per-project allocation currently set to 20,000 service units for 6 months. Allocations can be renewed or extended (see [Project and Allocation Management](#) section for more details on Chameleon allocations.)

Lease: To ensure fairness to all users, resource reservations (leases) are limited to a duration of 7 days. However, an active lease within 48 hours of its end time can be prolonged by up to 7 days from the moment of request if resources are available. To prolong a lease, click on the "Update Lease" button in the Reservations panel of the CHI OpenStack dashboard, and enter the additional duration requested in the "Prolong for" box including the unit suffix, e.g. "5d" for 5 days or "30m" for 30 minutes. If there is an advance reservation blocking your lease prolongation that could potentially be moved, you can interact through the users mailing list to coordinate with others users. Additionally, if you know from the start that your lease will require longer than a week and can justify it, you can [contact Chameleon staff via the ticketing system](#) to request a one-time exception to create a longer lease. The lease must be requested by the PI.

20.5.1 Service Units

Chameleon allocations can consist of several components of the system. Users can request allocation of individual compute nodes, storage servers, or complete Scalable Compute Units (SCUs) which contain compute servers, storage nodes, and an open flow switch.

Compute servers are allocated in Service Units (SUs), which equates to one hour of wall clock time on a single server (for virtual machines, an SU is 24 cores with up to 128GB of RAM). Note this unit differs from traditional HPC or cloud service units that are charged in core-hours; a Chameleon SU is a full server, as the type of experiments and performance measurements users may wish to do may be contaminated by sharing nodes.

Storage servers are also charged in SUs, at 2x the rate of compute servers (i.e., 1 hour allocation of 1 storage server == 2 SUs). SCUs are charged at the rate of 50 SUs per wall clock hour (42 compute servers, 4 storage nodes, plus one OpenFlow switch).

An allocation may make use of multiple SCUs, up to the size of the full testbed.

For example, a user wishing to provision a 10 node cluster +1 storage server for a 1 week experiment should budget $[(10 + 2) \text{ SUs per hour}] * [7 \text{ days} * 24 \text{ hours/day}] = 2,016 \text{ SUs}$ for that experiment.

SUs are charged the same regardless of use case. Hence, whether asking for bare metal access, virtual machine access, or use of default images, the charge is the same --- you are charged for the fraction of the resource your experiment occupies, regardless of the type of the experiment.

The basic principle for charging service units for Chameleon resources is to evaluate the amount of time a fraction of the resource is unavailable to other users. If a reservation is made through the portal for a particular date/time in the future, the user will be charged for this time regardless of whether the reservation is actually used, as the Chameleon scheduling system will have to drain the appropriate part of the system to satisfy the reservation, even if the nodes requested are not actually used. A reservation request may be cancelled in which case no charges will apply.

20.5.2 Project Allocation Size

CUrrently Chameleon is operating on a “soft allocation model” where each project, if approved, will receive a startup allocation of 20,000 SUs for six months that can be both recharged (i.e., more SUs can be added) and renewed (i.e., the duration can be extended) via submitting a renew/recharge request. This startup allocation value has been designed to respond to both PI needs (i.e., cover an amount of experimentation needed to obtain a significant result) and balance fairness to other users (it represents roughly 1% of testbed six months’ capacity). Requests for these startup projects will receive a fast track internal review (i.e., users can expect them to be approved within a few days).

A PI can apply for multiple projects/allocations; however, the number of held allocations will be taken into account during review.

As our understanding of user need grows we expect the Chameleon allocation model to evolve towards closer reflection of those needs in the form of more differentiated allocations that will allow us to give larger allocations to users for longer time.

Indiana University

Please be mindful to shutting down your VMS when not in use as even VMs that do not do any calculations get charged. In past classes we had students that did not shut down their VMs and within 2 weeks used up all SUs for the entire class of 70 students. We like to avoid this. In future cases we will assign the grade “F” to such students, as is customary also in other universities.

20.6 OpenStack Virtual Machines

OpenStack is an Infrastructure as a Service (IaaS) platform that allows you to create and manage virtual environments. Chameleon provides an installation of OpenStack version 2015.1 (Kilo) using the KVM virtualization technology.

Since the KVM hypervisor is used on this cloud, any virtual machines you upload must be compatible with KVM.

This tutorial provide basic information about how to use the OpenStack web interface and provides some information specific to using OpenStack KVM on Chameleon.

20.6.1 Web Interface

An easy way to use OpenStack KVM on Chameleon is via the [OpenStack web interface](#) also known as Horizon. You log into the web interface using your Chameleon username and password. If you change your Chameleon password in the portal, that change will propagate to the OpenStack KVM interface in about 5 minutes.

The initial log in page appears as:



After a successful log in, you will see the Overview page as shown below. This page provides a summary of your current and recent usage and provides links to various other pages. Most of the tasks you will perform are done via the menu on the lower left and will be described below. One thing to note is that on the left, your current project is displayed. If you have multiple Chameleon projects, you can change which of them is your current project. All of the information displayed and actions that you take apply to your current project. So in the screen shot below, the quota and usage apply to the current project you have selected and no information about your other projects is shown.

Limit Summary

Resource	Used	Limit
Instances	Used 0 of 16	16
VCPUs	Used 0 of 16	16
RAM	Used 0 Bytes of 31.3GB	31.3GB
Floating IPs	Used 0 of 8	8
Security Groups	Used 1 of 10	10
Volumes	Used 1 of 10	10
Volume Storage	Used 5GB of 100GB	100GB

Usage Summary

Select a period of time to query its usage:

From: 2014-12-01 To: 2014-12-10 The date should be in YYYY-mm-dd format.

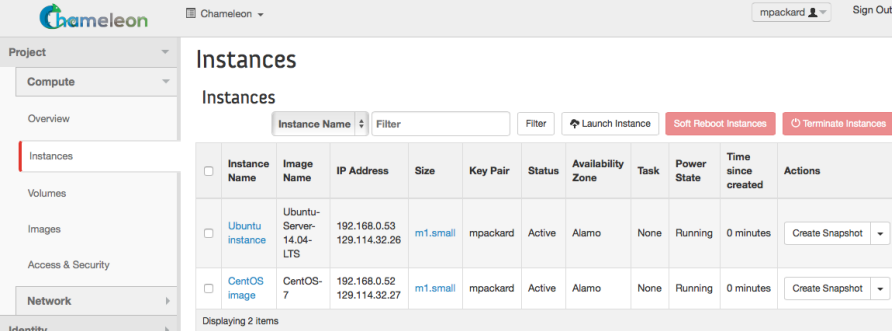
Active Instances: 0 Active RAM: 0 Bytes This Period's VCPU-Hours: 188.01 This Period's GB-Hours: 1504.19

Usage

Instance Name	VCPUs	Disk	RAM	Time since created
No items to display.				
Displaying 0 items				

Managing Virtual Machine Instances

One of the main activities you'll be performing in this web interface is the management of virtual machines, or instances. You do this via the Instances page that is reachable from the menu in the lower left of the Overview page. An example Instances page is shown below. For instances that you have running, you can click on the name of the instance to get more information about it and to access the VNC interface to the console. The dropdown menu to the left of the instance lets you perform a variety of tasks such as suspending, terminating, or rebooting the instance.



The screenshot shows the Chameleon Cloud web interface. The top navigation bar includes the Chameleon logo, the project name 'Chameleon', and a user profile 'mpackard' with a 'Sign Out' button. A left sidebar contains a 'Project' menu with options: Compute, Overview, Instances (highlighted), Volumes, Images, Access & Security, Network, and Identity. The main content area is titled 'Instances' and features a search bar with 'Instance Name' and 'Filter' dropdowns, and buttons for 'Launch Instance', 'Soft Reboot Instances', and 'Terminate Instances'. Below this is a table with the following data:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
Ubuntu-Server-14.04-LTS	Ubuntu-Server-14.04-LTS	192.168.0.53 129.114.32.26	m1.small	mpackard	Active	Alamo	None	Running	0 minutes	Create Snapshot
CentOS-7	CentOS-7	192.168.0.52 129.114.32.27	m1.small	mpackard	Active	Alamo	None	Running	0 minutes	Create Snapshot

At the bottom of the table, it says 'Displaying 2 items'.

The Instances page also lets you create new virtual machines by using the 'Launch Instance' button in the upper-right. When you click this button, a dialog window pops up. In the first 'Details' tab, you select the 'Instance Boot Source' of the instance, which is either an 'Image', a 'Snapshot' (an image created from a running virtual machine), or a 'Volume' (a persistent virtual disk that can be attached to a virtual machine). If you select 'Boot from image', the Image Name dropdown presents a list of virtual machine images that we have provided, that other Chameleon users have uploaded and made public, or images that you have uploaded for yourself. If you select 'Boot from snapshot', the Instance Snapshot dropdown presents a list of virtual machine images that you have created from your running virtual machines.

On the Details tab, you also provide a name for this instance (to help you identify instances that you are running), and select the amount of resources (Flavor) to allocate to the instance. If you select different flavors from the Flavor dropdown, their characteristics are displayed on the right.

Launch Instance

x

Details *
Access & Security *
Networking *
Post-Creation *
Advanced Options

Availability Zone

Alamo

Instance Name *

CentOS image

Flavor * ?

m1.small

Some flavors not meeting minimum image requirements have been disabled.

Instance Count * ?

1

Instance Boot Source * ?

Boot from image

Image Name

CentOS-7 (329.2 MB)

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.small
VCPUs	1
Root Disk	20 GB
Ephemeral Disk	0 GB
Total Disk	20 GB
RAM	2,048 MB

Project Limits

Number of Instances 2 of 16 Used

Number of VCPUs 2 of 16 Used

Total RAM 4,096 of 32,000 MB Used

Cancel Launch

The next tab is 'Access & Security', where you select an SSH keypair that will be inserted into your virtual machine. These keypairs can be uploaded via the main 'Access & Security' section. You will need to select a keypair here to be able to access an instance created from one of the public images Chameleon provides. These images are not configured with a default root password and you will not be able to log in to them without configuring an SSH key.

Launch Instance

x

Details *
Access & Security *
Networking *
Post-Creation *
Advanced Options

Key Pair ?

mpackard

Security Groups * ?

default

Control access to your instance via key pairs, security groups, and other mechanisms.

Cancel Launch

Next is 'Networking', where you select which network should be associated with the instance. Click the + next to your your project's private network (PROJECT_NAME-net), not ext-net.

Launch Instance

Details * Access & Security * **Networking *** Post-Creation * Advanced Options

Selected networks

NIC:1 Chameleon-net (13506b01-04f3-4f7d-8e26-b7bd287f3d5e) -

Available networks

ext-net (f275a55f-75f2-4667-a3af-3d40192ae385) +

Choose network from Available networks to Selected networks by push button or drag and drop, you may change NIC order by drag and drop as well.

Cancel Launch

Once you do this, you can Launch your instance and the Instances page will show progress as it starts.

If you would like to assign a public IP address to your VM, you can do that while it is booting up. Click on the dropdown under *Actions* and choose *Associate Floating IP*. Choose an IP from the *IP Address* menu and click *Associate*. If there are no addresses available, click the + and follow the prompts to add one.

Manage Floating IP Associations

IP Address *

IP Address *

129.114.32.32 +

Port to be associated *

test: 192.168.0.57

Select the IP address you wish to associate with the selected instance.

Cancel Associate

OpenStack injects your SSH key into the VM and you can use the corresponding private SSH key to log in to the VM. You will need to use the public IP assigned to your VM to connect from outside of Chameleon, or connect through an existing instance that both a public and private IP.

Note that the images we provide do not allow SSH into the root account. For root access, SSH into the instance as user 'cc' and then use the *sudo* command to become root.

We have enabled auto-login for the cc user on the console of our supported images. This should aid in debugging if you are unable to reach the instance via ssh for some reason.



Snapshots

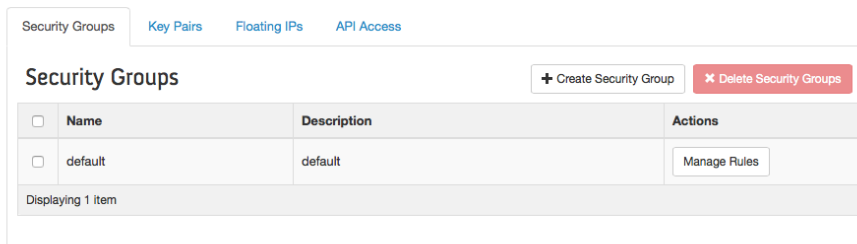
The instance list page shown above has an option ‘Create Snapshot’ that allows you to save a copy of the disk contents of a running virtual machine. This allows you to start new virtual machines in the future that are identical to this one and is an easy way to save any changes you make to a running virtual machine.

Firewall (Access Security)

Each project has control over their own firewall settings for their instances. At minimum you’ll probably want to allow SSH access so you can reach your instances.

To enable this traffic, you need to configure the security group used by your virtual machine. You can see a list of your security groups using the ‘‘Access & Security’’ link on the left.

Access & Security



To edit a security group, click on ‘‘Edit Rules’’. This opens a page showing the existing rules in the security group.

Manage Security Group Rules: default

Security Group Rules + Add Rule ✖ Delete Rules

<input type="checkbox"/>	Direction	Ether Type	IP Protocol	Port Range	Remote	Actions
<input type="checkbox"/>	Ingress	IPv4	Any	-	default	Delete Rule
<input type="checkbox"/>	Egress	IPv6	Any	-	:::0 (CIDR)	Delete Rule
<input type="checkbox"/>	Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)	Delete Rule
<input type="checkbox"/>	Ingress	IPv6	Any	-	default	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	ICMP	-	0.0.0.0/0 (CIDR)	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0 (CIDR)	Delete Rule

Displaying 6 items

Click on “Add Rule” and choose the *SSH* rule from the list, and click *Add*. Modifications are automatically propagated to the OpenStack cloud. Feel free to add other rules as necessary.

Add Rule ✕

Rule *

SSH

Remote * ?

CIDR

CIDR ?

0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the “Port Range” option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel Add

20.6.2 OpenStack REST Interfaces

The OpenStack REST Interfaces are supported on Chameleon over secure HTTP connections. You can download your OpenStack credentials file from the web interface via the “Access & Security” link in the left of any page and then click on the “API Access” link on the top.

You can then install the OpenStack command line clients following [these instructions](#). If using pip, we recommend setting up a virtualenv.

The SSL certificate used by Chameleon is trusted by most operating systems, so you shouldn’t have to provide any extra options to OpenStack commands, i.e. “nova list” should work. If your command-line tool complains about the certificate, [download the Mozilla CA bundle from the cURL website](#) and run the OpenStack client tools with the `--os-cacert cacert.pem` arguments.

20.6.3 Downloading and uploading data

You can use the OpenStack command line clients to download data from and upload data to Chameleon clouds. Configure your environment by following the “OpenStack REST Interfaces” section above, then use the following commands:

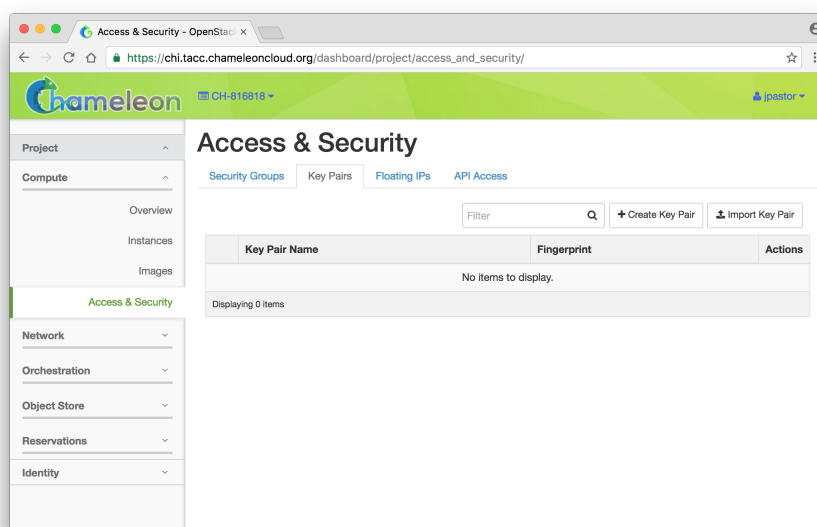
- `glance image-download` to download images and snapshots from Glance
- `glance image-create` to upload images and snapshots to Glance
- `cinder upload-to-image` to convert a Cinder volume to a Glance image
- `cinder create [--image-id <image-id>] [--image <image>]` to create a Cinder volume from a Glance image

20.7 Horizon Graphical User Interface

20.7.1 Configure resources

Once your lease is started, you are almost ready to start an instance. But first, you need to make sure that you will be able to connect to it by setting up a key pair. This only has to be done once per user per project.

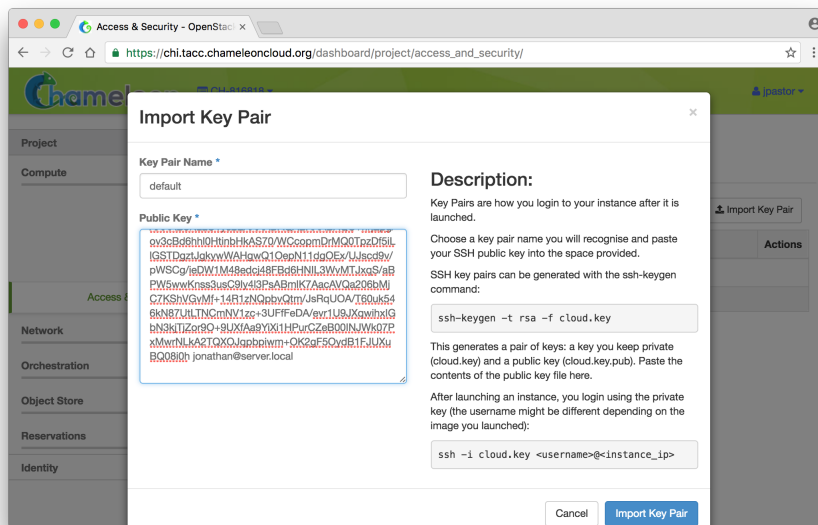
Go to Project > Compute > Access & Security, then select the Key Pairs tab.



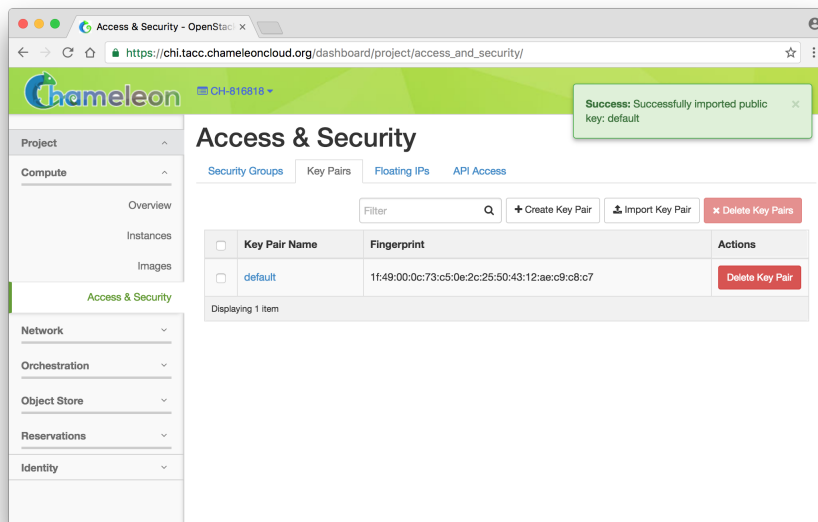
Here you can either ask OpenStack to create an SSH key pair for you (via the “Create Key” Pair button), or, if you already have an SSH key pair on your machine and are happy to use it, click on “Import Key Pair”.

If you chose to import a key pair, you will be asked to enter a name for the key pair, for example laptop. In the “Public Key” box, copy the content of your SSH public key. Typically it will be at `~/ssh/id_rsa.pub`. On Mac OS X, you can run in a terminal: `cat ~/ssh/id_rsa.pub | pbcopy`

It copies the content of the public key to your copy/paste buffer. Then you can simply paste in the “Public Key” box.

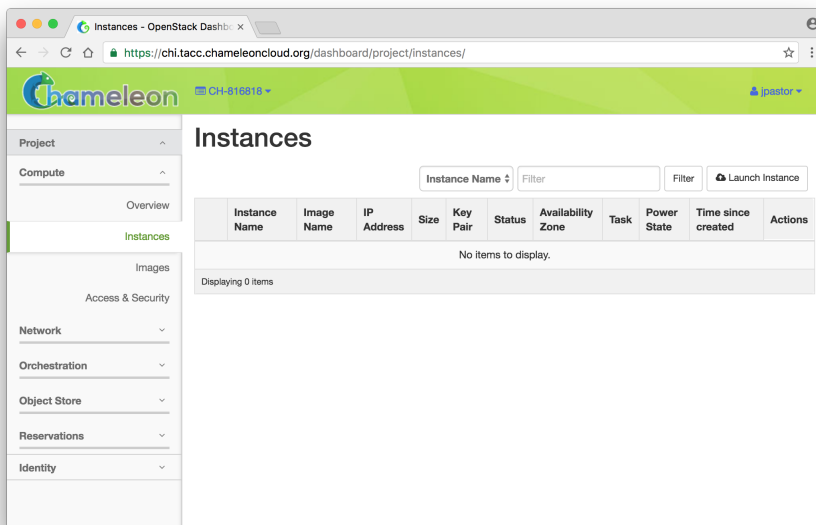


Then, click on the blue “Import Key Pair” button. This should show you the list of key pairs, with the one you just added.

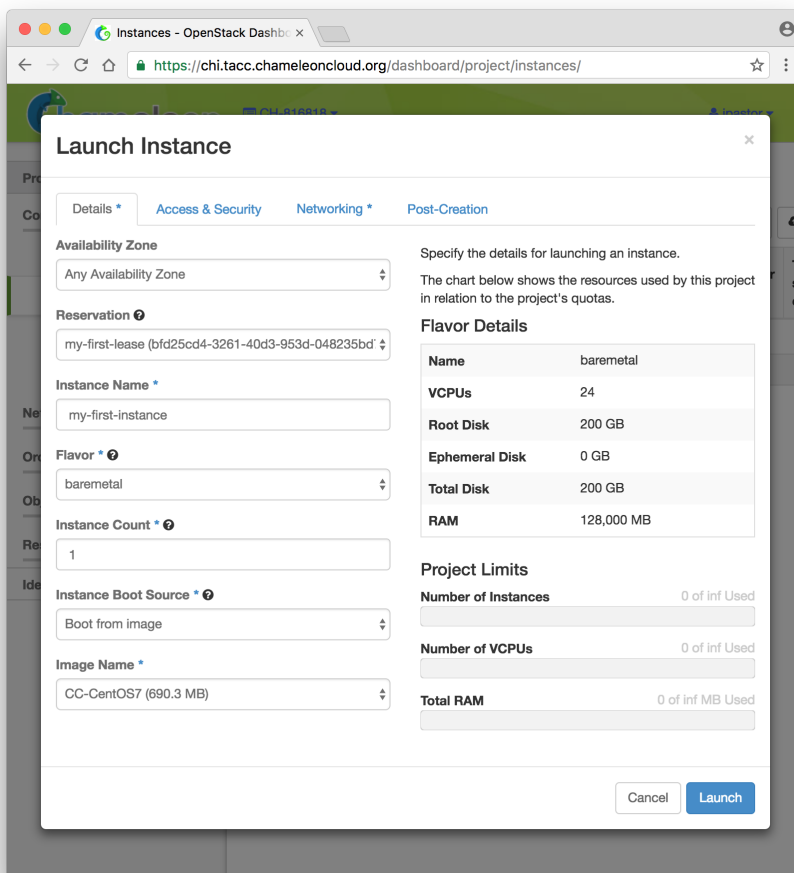


For those already familiar with OpenStack, note that Security Groups are not functional on bare-metal. All instances ports are open to the Internet and any security group rule you add will not be respected.

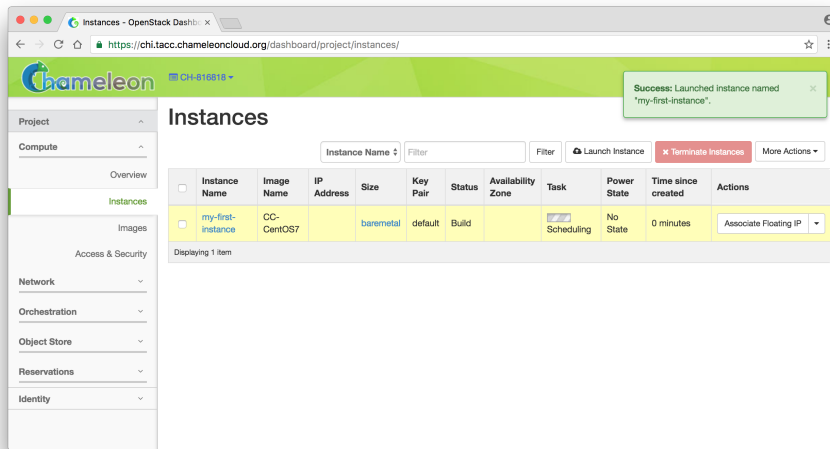
Now, go to the “Instances” panel.



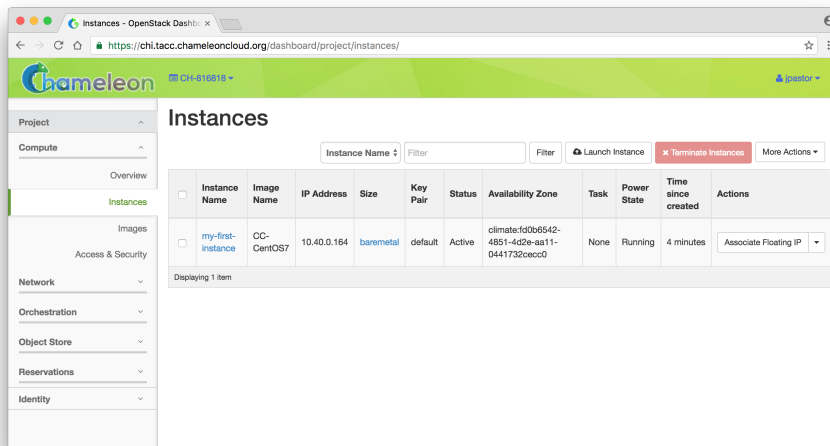
Click on the “Launch Instance” button in the top right corner. Select a reservation in the Reservation box, pick an instance name (in this example my-first-instance) and in the Image Name list select our default environment named CC-CentOS7. If you have multiple key pairs registered, you need to select one in the “Access & Security” tab. Finally, click on the blue “Launch” button.



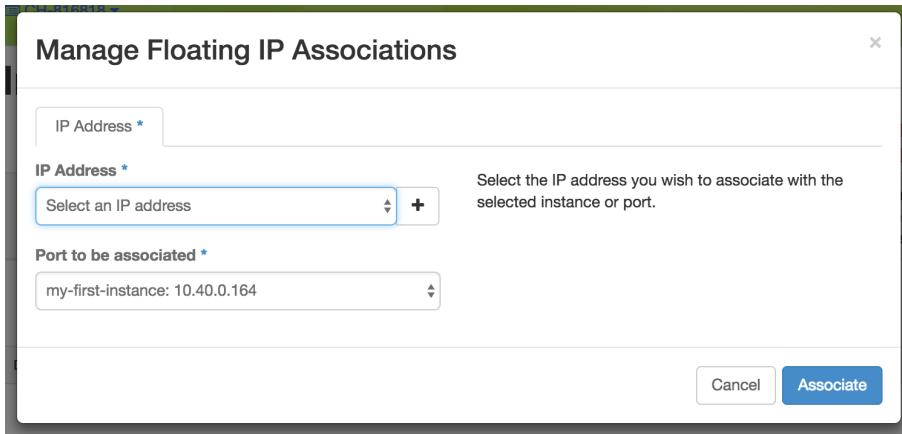
The instance will show up in the instance list, at first in Build status. It takes a few minutes to deploy the instance on bare-metal hardware and reboot the machine.



After a few minutes the instance should become in Active status and the Power State should be Running.



At this point the instance might still be booting: it might take a minute or two to actually be accessible on the network and accept SSH connections. In the meantime, you can attach a floating IP to the instance. Click on the "Associate Floating IP" button. You should get a screen like the one below:



Manage Floating IP Associations

IP Address *

IP Address *

Select an IP address +

Select the IP address you wish to associate with the selected instance or port.

Port to be associated *

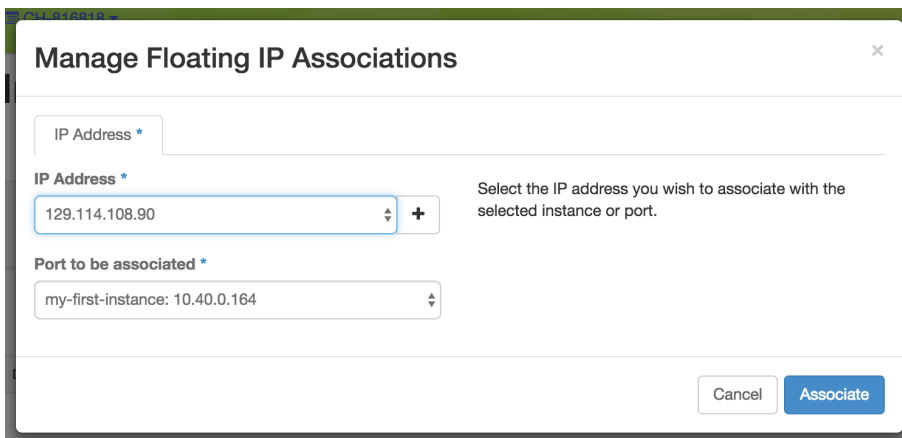
my-first-instance: 10.40.0.164

Cancel Associate

If there are no unused floating IP already allocated to your project, click on the + button. In the window that opens, select the ext-net pool if not already selected by default and click on the blue Allocate IP button.



You will be returned to the previous window. The correct value for “Port to be associated” should already be selected, so you only have to click on “Associate”.



Manage Floating IP Associations

IP Address *

IP Address *

129.114.108.90 +

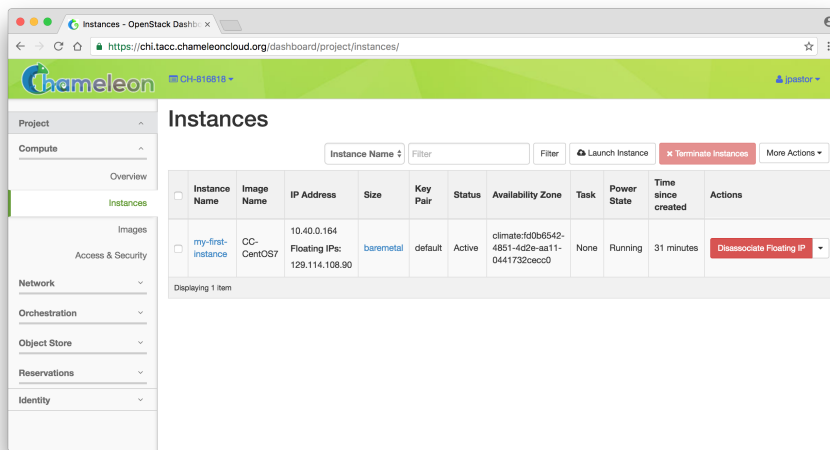
Select the IP address you wish to associate with the selected instance or port.

Port to be associated *

my-first-instance: 10.40.0.164

Cancel Associate

This should send you back to the instance list, where you can see the floating IP attached to the instance (you may need to refresh your browser to see the floating IP).



20.7.2 Interact with resources

Now you should be able to connect to the instance via SSH using the `cc` account. In a terminal, type `ssh cc@<floating_ip>`, in our example this would be `ssh cc@130.202.88.241`

SSH will probably tell you:

```
The authenticity of host \textquotesingle{}130.202.88.241
(130.202.88.241) can't be established. RSA key fingerprint
is 5b:ca:f0:63:6f:22:c6:96:9f:c0:4a:d8:5e:dd:fd:eb.
Are you sure you want to continue connecting (yes/no)?
```

Type `yes` and press `Enter`. You should arrive to a prompt like this one:

```
[cc@my-first-instance ~]$
```

If you notice SSH errors such as connection refused, password requests, or failures to accept your key, it is likely that the physical node is still going through the boot process. In that case, please wait before retrying. Also make sure that you use the `cc` account. If after 10 minutes you still cannot connect to the machine, please [open a ticket with our help desk](#).

You can now check whether the resource matches its known description in the resource registry. For this, simply run: `sudo cc-checks -v`

Warning

As of 03/30/2018, the `cc-checks` command may not work on the images in Chameleon cloud. You may have to ignore (not run) this command.


```

cc@test-new-image:~$ sudo cc-checks
Chassis
OK should have the correct serial number
OK should have the correct manufacturer
OK should have the correct product name

Disk
OK should have the correct name
OK should have the correct size
OK should have the correct model
OK should have the correct revision
OK should have the correct vendor

Virtual Hardware
OK should have the good driver

Memory
OK should have the correct size

Network
OK should be the correct interface name
OK should have the correct Driver
OK should have the correct Mac Address
OK should have the correct Rate
OK should have the correct version
OK should have the correct mounted mode
OK should not be a management card
OK should be the correct interface name
OK should have the correct Driver
OK should have the correct Mac Address
OK should have the correct Rate
OK should have the correct version
OK should have the correct mounted mode
OK should not be a management card

OS
OK should be the correct name
OK should be the correct kernel version
OK should be the correct version

Processor
OK should have the correct frequency
OK should be of the correct instruction set
OK should be of the correct model
OK should be of the correct version
OK should have the correct vendor
OK should have the correct description
OK should have the correct L1
OK should have the correct L1d
OK should have the correct L2
OK should have the correct L3

Rights on /tmp
OK should have mode 41777
[cc@test-new-image ~]$ echo $?
0
[cc@test-new-image ~]$

```

The `cc-checks` program prints the result of each check in green if it is successful and red if it failed. You can now run your experiment directly on the machine via SSH. You can run commands with root privileges by prefixing them with `sudo`. To completely switch user and become root, use the `sudo su - root` command.

Snapshot an instance

All instances in Chameleon, whether KVM or bare-metal, are running off disk images. The content of these disk images can be snapshotted at any point in time, which allows you to save your work and launch new instances from updated images later.

While OpenStack KVM has built-in support for snapshotting in the Horizon web interface and via the command line, bare-metal instances require a more complex process. To make this process easier, we developed the `cc-snapshot` tool, which implements snapshotting a bare-metal instance from command line and uploads it to Glance, so that it can be immediately used to boot a new bare-metal instance. The snapshot images created with this tool are whole disk images.

For ease of use, `cc-snapshot` has been installed in all the appliances supported by the Chameleon project. If you would like to use it in a different setting, it can be downloaded and installed from

the [github repository](#).

Once `cc-snapshot` is installed, to make a snapshot of a bare-metal instance, run the following command from inside the instance:

```
sudo cc-snapshot <snapshot_name>
```

You can verify that it has been uploaded to Glance by running the following command:

```
glance image-list
```

If you prefer to use a series of standard Unix commands, or are generally interested in more detail about image management, please refer to our [image management guide](#).

20.7.3 Use FPGAs

Consult the [dedicated page](#) if you would like to use the FPGAs available on Chameleon.

20.7.4 Next Step

Now that you have created some resources, it is time to interact with them! You will find instructions to the next step by visiting the following link:

- [Monitor resources and collect results](#)

20.8 HEAT

Deploying an MPI cluster, an OpenStack installation, or any other type of cluster in which nodes can take on multiple roles can be complex: you have to provision potentially hundreds of nodes, configure them to take on various roles, and make them share information that is generated or assigned only at deployment time, such as hostnames, IP addresses, or security keys. When you want to run a different experiment later you have to redo all this work. When you want to reproduce the experiment, or allow somebody else to reproduce it, you have to take very precise notes and pay great attention to their execution.

To help solve this problem and facilitate reproducibility and sharing, the Chameleon team configured a tool that allows you to deploy complex clusters with “one click”. This tool requires not just a simple image (i.e., appliance) but also a document, called a template, that contains the information needed to orchestrate the deployment and configuration of such clusters. We call this image + template combination complex appliance because it consists of more than just the image (i.e., appliance).

20.8.1 Supporting Complex Appliances

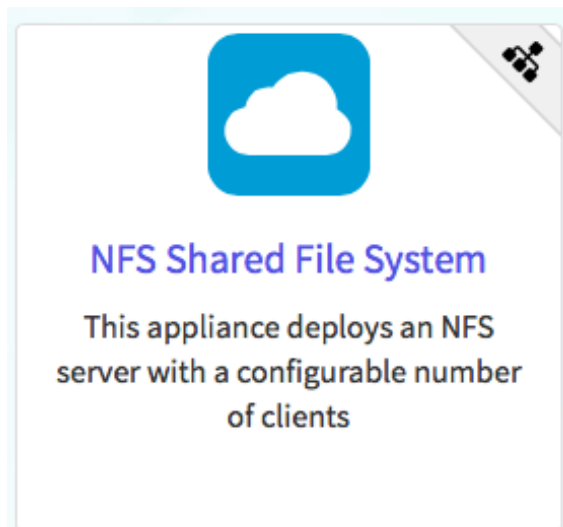
In a nutshell, complex appliances allow you to specify not only what image you want to deploy but also on how many nodes you want to deploy that image, what roles the deployed instances should boot into (such as e.g., head node and worker node in a cluster), what information from a specific instance should be passed to another instance in that complex appliance, and what scripts should be executed on boot so that this information is properly used for configuring the “one click” cluster. For example, a Network File System (NFS) appliance that we will use as an example in this guide, might specify deployment on three nodes, out of which one will be configured as head node and

others as worker nodes, the information passed between the images will be hostname of the head node, and the scripts executed on the worker nodes on boot will put that hostname in the fstab file. As you can tell from this description, images used for complex appliances are typically configured such that they can be booted into any role required on the one-click cluster we are booting; in this case the image will have both the software for NFS server node and client node.

Since complex appliances in Chameleon are currently implemented using the [OpenStack Heat](#) orchestration service, we will be using OpenStack terminology and features to work with them. The templates described above are YAML files using the [Heat Orchestration Template \(HOT\) format](#) (Heat also supports the AWS CloudFormation template format, but this is not covered here). A deployed complex appliance is referred to as a “stack” -- just as a deployed single appliance is typically referred to as an “instance”. This guide will tell you all you need to know in order to use and configure complex appliances on Chameleon; if you would like to know more about Heat, please refer to its [official documentation](#).

20.8.2 Chameleon Appliance Catalog

Our [Appliance Catalog](#) has several complex appliances for popular technologies that people want to deploy such as OpenStack or MPI or even more advanced deployments such as efficient SR-IOV enabled MPI in KVM virtual machines. We also provide common building blocks for cluster architectures, such as an NFS share. Complex appliances are identified by a badge in their top-right corner representing a group of machines, as shown in the screenshot:



20.8.3 Deployment

We will explain how to launch a complex appliance based on our [NFS share appliance](#). To launch a complex appliance, you only need to follow these steps:

1. Create a lease: use the OpenStack web interface (choose between CHI@UC or CHI@TACC) to create a lease. To launch our NFS appliance, reserve at least three compute nodes (the strict minimum is two nodes but we will use three in this example and later ones).
2. Go to the [Appliance Catalog](#) and identify the appliance you want to launch. In our case you can go straight to the [NFS share appliance](#); click on it to open its details page. You will see a “Launch” button and a “Get Template” button. Follow the “Get Template” link and copy its url to the clipboard -- you will need it in the following steps.

3. Click on the “Launch Complex Appliance at CHI@TACC” or “Launch Complex Appliance at CHI@UC” button depending on where your reservation was created.

This will take you to the Stacks page within the Orchestration menu. This page will show the current list of stacks, with controls to manage them and create new ones. Since we haven’t launched any yet, this list will be empty for now.

We will now create a new stack, which corresponds to the launch of a template. Click on Launch Stack on the top right. A window will pop up like below:

Select Template ×

Template Source *

File

Template File ?

Choose File no file selected

Environment Source

File

Environment File ?

Choose File no file selected

Description:

Use one of the available template source options to specify the template to be used in creating this stack.

Cancel
Next

We will deploy the NFS appliance described earlier; it will consist of a server node and two client nodes. Change the template source field to URL, and paste the URL of the [NFS share template](#) (if you don’t have it in your clipboard anymore you will need to go back to the appliance and get it by clicking on “Get template” again).

Don’t change the environment source settings, and click “Next”.

The next screen allows your to enter input values to your Heat template. Choose a name for your stack (e.g. my-nfs-cluster). Ignore the “Creation Timeout” and “Rollback On Failure” settings. You also need to enter your Chameleon password. Then, you need to select a value for the three parameters of the template: for key_name, choose your SSH key pair (this key pair will authorize access on each deployed instances, both server and client). For nfs_client_count, change the default value of 1 to 2. For reservation_id, choose your reservation created earlier. Finally, click “Launch”.

Launch Stack ×

Stack Name * ?

Description:

Create a new stack with the provided values.

Creation Timeout (minutes) * ?

Rollback On Failure ?

Password for user "priteau" * ?

key_name ?

nfs_client_count ?

reservation_id * ?

Your stack should be in status “Create In Progress” for several minutes while it first launches the NFS server instance, followed by the NFS client instances.

<input type="checkbox"/>	Stack Name	Created	Updated	Status	Actions
<input type="checkbox"/>	my-nfs-cluster	0 minutes	Never	Create In Progress	Check Stack ▾

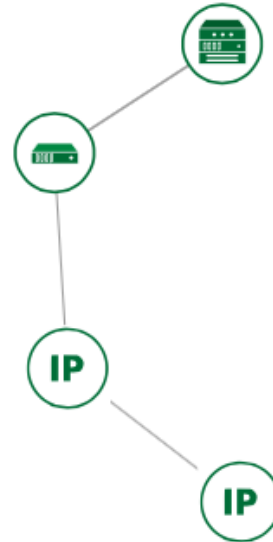
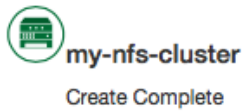
Displaying 1 item

It will then move to the status “Create Complete”.

<input type="checkbox"/>	Stack Name	Created	Updated	Status	Actions
<input type="checkbox"/>	my-nfs-cluster	15 minutes	Never	Create Complete	Check Stack ▾

Displaying 1 item

You can click on the stack name to get more details, including a visualization of the deployed resources, as pictured below. The single machine inside a circle represents the NFS server instance. The rack of machine represents the group of NFS client instances (in this case, a group composed of two instances). The server’s floating IP (the public IP assigned to a resource) is represented by an IP in a circle; an IP in a circle is also used to represent the association of the IP with the NFS server instance (not the greatest idea to use the same symbol for both the IP and the association -- we agree but can’t do much about it at the moment). Blow off some steam by dragging the visualization across the screen, it can be rather fun!



You can now ssh to the server using the floating IP just as you do with regular instances (use the cc account). The client does not have a floating IP attached to it (as per the visualization above) but you can connect to it via the server node with the client’s private IP (connect to the server with `ssh -A` to enable the SSH agent forwarding after loading your key to your SSH agent with `ssh-add <path-to-your-key>`).

You can find out the information about the IPs and other things if you click the “Overview” tab and look in the “Outputs” section. Under the “Resources” tab you will see the resources described above (the server, clients, server’s public/floating IP, and its the association) and information about them. In the “Events” tab you will see information about the history of the deployment so far. In Template you will see the template that was used to deploy this stack.

20.8.4 Heat Template

The NFS share appliance deploys:

- an NFS server instance, that exports the directory `/exports/example` to any instance running on Chameleon bare-metal,
- one or several NFS client instances, which configure `/etc/fstab` to mount this NFS share to `/mnt` (and can subsequently read from and write to it).

This template is reproduced further below, and includes inline comments starting with the `#` character. There are three main sections:

- resources,
- parameters,
- outputs.

The resources section is the most important part of the template: it defines which OpenStack resources to create and configure. Inside this section you can see four resources defined:

- `nfs_server_floating_ip`

- `nfs_server`
- `nfs_server_ip_association`
- `nfs_clients`

The first resource, `nfs_server_floating_ip`, creates a floating IP on the ext-net public network. It is not attached to any instance yet.

The second resource, `nfs_server`, creates the NFS server instance (an instance is defined with the type `OS::Nova::Server` in Heat). It is a bare-metal instance (`flavor: baremetal`) using the CC-CentOS7 image and connected to the private network named `sharednet1`. We set the keypair to use the value of the parameter defined earlier, using the `get_param` function. Similarly, the reservation to use is passed to the scheduler. Finally, a user-data script is given to the instance, which configures it as an NFS server exporting `/exports/example` to Chameleon instances.

The `nfs_server_ip_association` resource associates the floating IP created earlier with the NFS server instance.

Finally, the `nfs_clients` resource defines a resource group containing instance configured to be NFS clients and mount the directory exported by the NFS server defined earlier. The IP of the NFS server is gathered using the `get_attr` function, and placed into user-data using the `str_replace` function.

Parameters all have the same data structure: each one has a name (`key_name` or `reservation_id` in this case), a data type (number or string), a comment field called `description`, optionally a default value, and a list of constraints (in this case only one per parameter). Constraints tell Heat to match a parameter to a specific type of OpenStack resource. Complex appliances on Chameleon require users to customize at least the key pair name and reservation ID, and will generally provide additional parameters to customize other properties of the cluster, such as its size, as in this example.

Outputs are declared similarly to parameters: they each have a name, an optional description, and a value. They allow to return information from the stack to the user.

```
# This describes what is deployed by this template.
description: NFS server and clients deployed with Heat on Chameleon

# This defines the minimum Heat version required by this template.
heat_template_version: 2015-10-15

# The resources section defines what OpenStack resources are to be deployed and
# how they should be configured.
resources:
  nfs_server_floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: ext-net

  nfs_server:
    type: OS::Nova::Server
    properties:
      flavor: baremetal
      image: CC-CentOS7
      key_name: { get_param: key_name }
      networks:
        - network: sharednet1
      scheduler_hints: { reservation: { get_param: reservation_id } }
    user_data: |
      #!/bin/bash
      yum install -y nfs-utils
```

```

mkdir -p /exports/example
chown -R cc:cc /exports
echo '/exports/example 10.140.80.0/22(rw,async) 10.40.0.0/23(rw,async)' >> /etc/exports
systemctl enable rpcbind && systemctl start rpcbind
systemctl enable nfs-server && systemctl start nfs-server

nfs_server_ip_association:
  type: OS::Nova::FloatingIPAssociation
  properties:
    floating_ip: { get_resource: nfs_server_floating_ip }
    server_id: { get_resource: nfs_server }

nfs_clients:
  type: OS::Heat::ResourceGroup
  properties:
    count: { get_param: nfs_client_count }
    resource_def:
      type: OS::Nova::Server
      properties:
        flavor: baremetal
        image: CC-CentOS7
        key_name: { get_param: key_name }
        networks:
          - network: sharednet1
      scheduler_hints: { reservation: { get_param: reservation_id } }
      user_data:
        str_replace:
          template: |
            #!/bin/bash
            yum install -y nfs-utils
            echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
            mount -a
          params:
            $nfs_server_ip: { get_attr: [nfs_server, first_address] }

# The parameters section gathers configuration from the user.
parameters:
  nfs_client_count:
    type: number
    description: Number of NFS client instances
    default: 1
    constraints:
      - range: { min: 1 }
        description: There must be at least one client.
  key_name:
    type: string
    description: Name of a KeyPair to enable SSH access to the instance
    default: default
    constraints:
      - custom_constraint: nova.keypair
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
      - custom_constraint: blazar.reservation

outputs:
  server_ip:
    description: Public IP address of the NFS server
    value: { get_attr: [nfs_server_floating_ip, ip] }
  client_ips:

```



```
description: Private IP addresses of the NFS clients
value: { get_attr: [nfs_clients, first_address] }
```

20.8.5 Customizing an existing template

Customizing an existing template is a good way to start developing your own. We will use a simpler template than the previous example to start with: it is the [Hello World complex appliance](#).

First, delete the stack you launched, because we will need all three nodes to be free. To do this, go back to the Project > Orchestration > Stacks page, select your stack, and then click on the red “Delete Stacks” button. You will be asked to confirm, so click on the blue “Delete Stacks” button.

Confirm Delete Stacks ×

You have selected "my-nfs-cluster". Please confirm your selection. This action cannot be undone.

The template for the [Hello World complex appliance](#) is reproduced below. It is similar to the NFS share appliance, except that it deploys only a single client. You can see that it has four resources defined:

- nfs_server_floating_ip
- nfs_server
- nfs_server_ip_association
- nfs_client

The `nfs_client` instance mounts the NFS directory shared by the `nfs_server` instance, just like in our earlier example.

```
# This describes what is deployed by this template.
description: NFS server and client deployed with Heat on Chameleon

# This defines the minimum Heat version required by this template.
heat_template_version: 2015-10-15

# The resources section defines what OpenStack resources are to be deployed and
# how they should be configured.
resources:
  nfs_server_floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: ext-net

  nfs_server:
    type: OS::Nova::Server
    properties:
      flavor: baremetal
      image: CC-CentOS7
      key_name: { get_param: key_name }
      networks:
        - network: sharednet1
      scheduler_hints: { reservation: { get_param: reservation_id } }
      user_data: |
        #!/bin/bash
```

```

yum install -y nfs-utils
mkdir -p /exports/example
chown -R cc:cc /exports
echo '/exports/example 10.140.80.0/22(rw,async) 10.40.0.0/23(rw,async)' >> /etc/exports
systemctl enable rpcbind && systemctl start rpcbind
systemctl enable nfs-server && systemctl start nfs-server

nfs_server_ip_association:
  type: OS::Nova::FloatingIPAssociation
  properties:
    floating_ip: { get_resource: nfs_server_floating_ip }
    server_id: { get_resource: nfs_server }

nfs_client:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
    key_name: { get_param: key_name }
    networks:
      - network: sharednet1
    scheduler_hints: { reservation: { get_param: reservation_id } }
    user_data:
      str_replace:
        template: |
          #!/bin/bash
          yum install -y nfs-utils
          echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
          mount -a
      params:
        $nfs_server_ip: { get_attr: [nfs_server, first_address] }

# The parameters section gathers configuration from the user.
parameters:
  key_name:
    type: string
    description: Name of a KeyPair to enable SSH access to the instance
    default: default
    constraints:
      - custom_constraint: nova.keypair
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
      - custom_constraint: blazar.reservation

```

Download this template from the [Hello World complex appliance details page](#) to your local machine, and open it in your favorite text editor.

We will customize the template to add a second NFS client by creating a new resource called `another_nfs_client`. Add the following text to your template inside the resources section. Make sure to respect the level of indentation, which is important in YAML.

```

another_nfs_client:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
    key_name: { get_param: key_name }
    networks:
      - network: sharednet1

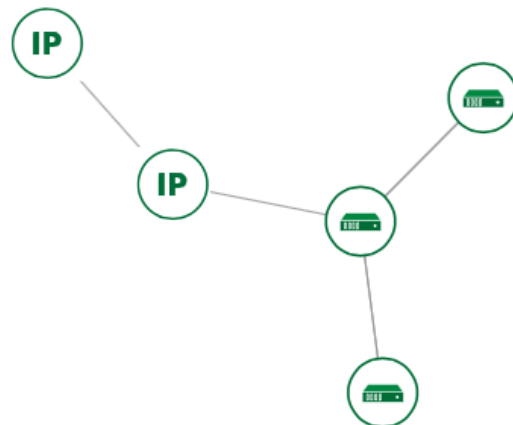
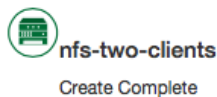
```

```

scheduler_hints: { reservation: { get_param: reservation_id } }
user_data:
  str_replace:
    template: |
      #!/bin/bash
      yum install -y nfs-utils
      echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
      mount -a
    params:
      $nfs_server_ip: { get_attr: [nfs_server, first_address] }

```

Now, launch a new stack with this template. Since the customized template is only on your computer and cannot be addressed by a URL, use the “Direct Input” method instead and copy/paste the content of the customized template. The resulting topology view is shown below: as you can see, the two client instances are shown separately since each one is defined as a separate resource in the template.



You may have realized already that while adding just one additional client instance was easy, launching more of them would require to copy / paste blocks of YAML many times while ensuring that the total count is correct. This would be easy to get wrong, especially when dealing with tens or hundreds of instances.

So instead, we leverage another construct from Heat: resource groups. Resource groups allow to define one kind of resource and request it to be created any number of times.

Remove the `nfs_client` and `another_client` resources from your customized template, and replace them with the following:

```

nfs_clients:
  type: OS::Heat::ResourceGroup
  properties:
    count: 2
    resource_def:
      type: OS::Nova::Server
      properties:

```

```

flavor: baremetal
image: CC-CentOS7
key_name: { get_param: key_name }
networks:
  - network: sharednet1
scheduler_hints: { reservation: { get_param: reservation_id } }
user_data:
  str_replace:
    template: |
      #!/bin/bash
      yum install -y nfs-utils
      echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
      mount -a
  params:
    $nfs_server_ip: { get_attr: [nfs_server, first_address] }

```

A resource group is configured with a properties field, containing the definition of the resource to launch (`resource_def`) and the number of resources to launch (`count`). Once launched, you will notice that the topology view groups all client instances under a single Resource Group icon. We use the same `resource_def` than when defining separate instances earlier.

Another way we can customize this template is by adding outputs to the template. Outputs allow a Heat template to return data to the user. This can be useful to return values like IP addresses or credentials that the user must know to use the system.

We will create an output returning the floating IP address used by the NFS server. We define an outputs section, and one output with the name `server_ip` and a description. The value of the output is gathered using the `get_attr` function which obtains the IP address of the server instance.

```

outputs:
  server_ip:
    description: Public IP address of the NFS server
    value: { get_attr: [nfs_server_floating_ip, ip] }

```

You can get outputs in the “Overview” tab of the Stack Details page. If you want to use the command line, install `python-heatclient` and use the `heat output-list` and `heat output-show` commands, or get a full list in the information returned by `heat stack-show`.

Multiple outputs can be defined in the outputs section. Each of them needs to have a unique name. For example, we can add another output to list the private IPs assigned to client instances:

```

client_ips:
  description: Private IP addresses of the NFS clients
  value: { get_attr: [nfs_clients, first_address] }

```

The image below shows the resulting outputs as viewed from the web interface. Of course IP addresses will be specific to each deployment.

Outputs

client_ips	Private IP address of the NFS clients ["10.140.82.20", "10.140.82.19"]
server_ip	Public IP address of the NFS server 130.202.88.157

Finally, we can add a new parameter to replace the hardcoded number of client instances by a value passed to the template. Add the following text to the parameters section:

```
nfs_client_count:
  type: number
  description: Number of NFS client instances
  default: 1
  constraints:
    - range: { min: 1 }
      description: There must be at least one client.
```

Inside the resource group definition, change `count: 2` to `count: { get_param: nfs_client_count }` to retrieve and use the parameter we just defined. When you launch this template, you will see that an additional parameter allows you to define the number of client instances, like in the NFS share appliance.

At this stage, we have fully recreated the NFS share appliance starting from the Hello World one! The next section will explain how to write a new template from scratch.

20.8.6 Writing a new template

You may want to write a whole new template, rather than customizing an existing one. Each template should follow the same layout and be composed of the following sections:

- Heat template version
- Description
- Resources
- Parameters
- Outputs

Heat template version

Each Heat template has to include the `heat_template_version` key with a valid version of HOT (Heat Orchestration Template). Chameleon bare-metal supports any HOT version up to 2015-10-15, which corresponds to OpenStack Liberty. The [Heat documentation](#) lists all available versions and their features. We recommended that you always use the latest supported version to have access to all supported features:

```
heat_template_version: 2015-10-15
```

Description

While not mandatory, it is good practice to describe what is deployed and configured by your template. It can be on a single line:

```
description: This describes what this Heat template deploys on Chameleon.
```

If a longer description is needed, you can provide multi-line text in YAML, for example:

```
description: >
  This describes what this Heat
  template deploys on Chameleon.
```

Resources

The resources section is required and must contain at least one resource definition. A [complete list of resources types known to Heat](#) is available.

However, only a subset of them are supported by Chameleon, and some are limited to administrative use. We recommend that you only use:

- OS::Glance::Image
- OS::Heat::ResourceGroup
- OS::Heat::SoftwareConfig
- OS::Heat::SoftwareDeployment
- OS::Heat::SoftwareDeploymentGroup
- OS::Neutron::FloatingIP
- OS::Neutron::FloatingIPAssociation
- OS::Neutron::Port (advanced users only)
- OS::Nova::Keypair
- OS::Nova::Server

If you know of another resource that you would like to use and think it should be supported by the OpenStack services on Chameleon bare-metal, please let us know via our help desk.

Parameters

Parameters allow users to customize the template with necessary or optional values. For example, they can customize which Chameleon appliance they want to deploy, or which key pair to install. Default values can be provided with the `default` key, as well as constraints to ensure that only valid OpenStack resources can be selected. For example, `custom_constraint: glance.image` restricts the image selection to an available OpenStack image, while providing a pre-filled selection box in the web interface. [More details about constraints](#) are available in the Heat documentation.

Outputs

Outputs allow template to give information from the deployment to users. This can include usernames, passwords, IP addresses, hostnames, paths, etc. The outputs declaration is using the following format:

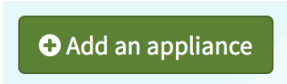
```
outputs:
  first_output_name:
    description: Description of the first output
    value: first_output_value
  second_output_name:
    description: Description of the second output
    value: second_output_value
```

Generally values will be calls to `get_attr`, `get_param`, or some other function to get information from parameters or resources deployed by the template and return them in the proper format to the user.

20.8.7 Sharing new complex appliances

If you have written your own complex appliances or substantially customized an existing one, we would love if you shared them with our user community!

The process is very similar to regular appliances: log into the Chameleon portal, go to the [appliance catalog](#), and click on the button in the top-right corner: “Add an appliance” (you need to be logged in to see it).


 A green rectangular button with rounded corners, containing a white plus sign followed by the text "Add an appliance".

You will be prompted to enter a name, description, and documentation. Instead of providing appliance IDs, copy your template to the dedicated field. Finally, share your contact information and assign a version string to your appliance. Once submitted, your appliance will be reviewed. We will get in touch if a change is needed, but if it's all good we will publish it right away!

20.8.8 Advanced topics

All-to-all information exchange

The previous examples have all used user-data scripts to provide instances with contextualization information. While it is easy to use, this contextualization method has a major drawback: because it is given to the instance as part of its launch request, it cannot use any context information that is not yet known at this time.

In practice, this means that in a client-server deployment, only one of these patterns will be possible:

- The server has to be deployed first, and once it is deployed, the clients can be launched and contextualized with information from the server. The server won't know about the clients unless there is a mechanism (not managed by Heat) for the client to contact the server.
- The clients have to be deployed first, and once they are deployed, the server can be launched and contextualized with information from the clients. The clients won't know about the server unless there is a mechanism (not managed by Heat) for the server to contact the clients.

This limitation was already apparent in our NFS share appliance: this is why the server instance exports the file system to all bare-metal instances on Chameleon, because it doesn't know which specific IP addresses are allocated to the clients.

This limitation is even more important if the deployment is not hierarchical, i.e. all instances need to know about all others. For example, a cluster with IP and hostnames populated in `/etc/hosts` required each instance to be known by every other instance.

This section presents a more advanced form of contextualization that can perform this kind of information exchange. This is implemented by Heat agents running inside instances and communicating with the Heat service to send and receive information. This means you will need to use an image bundling these agents. Currently, our CC-CentOS7 appliance and its CUDA version are the only ones supporting this mode of contextualization. If you build your own images using the [CC-CentOS7 appliance builder](#), you will automatically have these agents installed.

This contextualization is performed with several Heat resources:

- `OS::Heat::SoftwareConfig`. This resource describes code to run on an instance. It can be configured with inputs and provide outputs.
- `OS::Heat::SoftwareDeployment`. This resource applies a `SoftwareConfig` to a specific instance.
- `OS::Heat::SoftwareDeploymentGroup`. This resource applies a `SoftwareConfig` to a specific group of instances.

The template below illustrates how it works. It launches a group of instances that will automatically populate their `/etc/hosts` file with IP and hostnames from other instances in the deployment.

```
heat_template_version: 2015-10-15
```

```

description: >
  This template demonstrates how to exchange hostnames and IP addresses to populate /etc/hosts.

parameters:
  flavor:
    type: string
    default: baremetal
    constraints:
      - custom_constraint: nova.flavor
  image:
    type: string
    default: CC-CentOS7
    constraints:
      - custom_constraint: glance.image
  key_name:
    type: string
    default: default
    constraints:
      - custom_constraint: nova.keypair
  instance_count:
    type: number
    default: 2
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
      - custom_constraint: blazar.reservation

resources:
  export_hosts:
    type: OS::Heat::SoftwareConfig
    properties:
      outputs:
        - name: hosts
      group: script
      config: |
        #!/bin/sh
        (echo -n $(factor ipaddress); echo -n ' '; echo $(factor hostname)) > ${heat_outputs_path}.ho

  export_hosts_sdg:
    type: OS::Heat::SoftwareDeploymentGroup
    properties:
      config: { get_resource: export_hosts }
      servers: { get_attr: [server_group, refs_map] }
      signal_transport: HEAT_SIGNAL

  populate_hosts:
    type: OS::Heat::SoftwareConfig
    properties:
      inputs:
        - name: hosts
      group: script
      config: |
        #!/usr/bin/env python
        import ast
        import os
        import string
        import subprocess
        hosts = os.getenv('hosts')
        if hosts is not None:
            hosts = ast.literal_eval(string.replace(hosts, '\n', '\\n'))

```



```

        with open('/etc/hosts', 'a') as hosts_file:
            for ip_host in hosts.values():
                hosts_file.write(ip_host.rstrip() + '\n')

populate_hosts_sdg:
    type: OS::Heat::SoftwareDeploymentGroup
    depends_on: export_hosts_sdg
    properties:
        config: { get_resource: populate_hosts }
        servers: { get_attr: [server_group, refs_map] }
        signal_transport: HEAT_SIGNAL
        input_values:
            hosts: { get_attr: [ export_hosts_sdg, hosts ] }

server_group:
    type: OS::Heat::ResourceGroup
    properties:
        count: { get_param: instance_count }
        resource_def:
            type: OS::Nova::Server
            properties:
                flavor: { get_param: flavor }
                image: { get_param: image }
                key_name: { get_param: key_name }
                networks:
                    - network: sharednet1
                scheduler_hints: { reservation: { get_param: reservation_id } }
                user_data_format: SOFTWARE_CONFIG
                software_config_transport: POLL_SERVER_HEAT

outputs:
    deployment_results:
        value: { get_attr: [export_hosts_sdg, hosts] }

```

There are two SoftwareConfig resources.

The first SoftwareConfig, `export_hosts`, uses the `facter` tool to extract IP address and host-name into a single line (in the format expected for `/etc/hosts`) and writes it to a special path (`${heat_outputs_path}.hosts`). This prompts Heat to assign the content of this file to the output with the name `hosts`.

The second SoftwareConfig, `populate_hosts`, takes as input a variable named `hosts`, and applies a script that reads the variable from the environment, parses it with `ast.literal_eval` (as it is formatted as a Python dict), and writes each value of the dictionary to `/etc/hosts`.

The SoftwareDeploymentGroup resources `export_hosts_sdg` and `populate_hosts_sdg` apply each SoftwareConfig to the instance ResourceGroup with the correct configuration.

Finally, the instance ResourceGroup is configured so that each instance uses the following contextualization method instead of a user-data script:

```

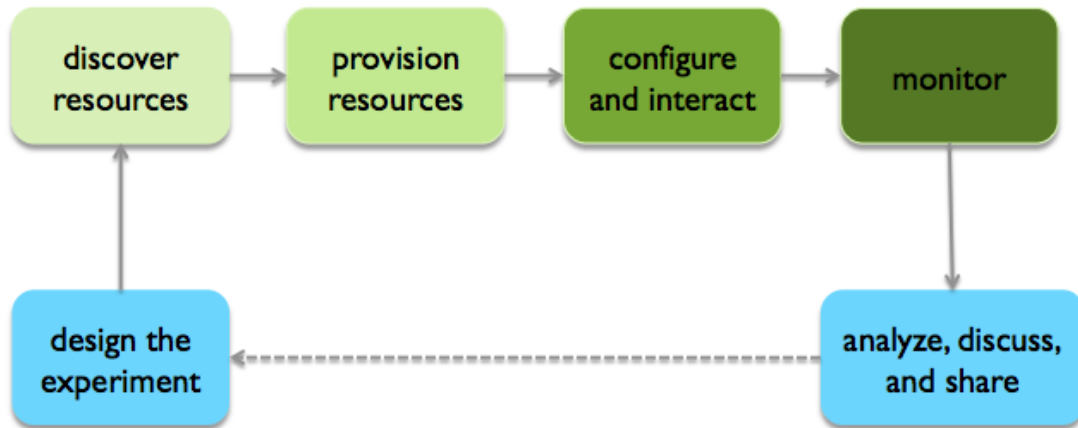
user_data_format: SOFTWARE_CONFIG
software_config_transport: POLL_SERVER_HEAT

```

You can follow the same template pattern to configure your own deployment requiring all-to-all information exchange.

20.9 Bare Metal

In this page you will find documentation guiding you through the bare-metal deployment features available in Chameleon. Chameleon gives users administrative access to bare-metal compute resources to run cloud computing experiments with a high degree of customization and repeatability. Typically, an experiment will go through several phases, as illustrated in the figure below:



The bare-metal user guide comes in two editions. The first is how to use Chameleon resources via the web interface, the recommended choice for new users to quickly learn how to use our testbed:

Get started with Chameleon using the web interface

1. [Discover Resources](#)
2. [Provision Resources](#)
3. [Configure and Interact](#)
4. [Monitor and Collect Results](#)

The second targets advanced users who are already familiar with Chameleon and would like to learn how to use Chameleon from the command line or with scripts.

Get started with Chameleon using the command line (advanced)

1. [Discover Resources](#)
2. [Provision Resources](#)
3. [Configure and Interact](#)
4. [Monitor and Collect Results](#)

You do not need to strictly follow the documentation sequentially. However, note that some steps assume that previous ones have been successfully performed.

You can also consult documentation describing how to use advanced features of Chameleon not covered by the guides above:

- the [Chameleon Object Store](#),
- [network isolation for bare metal](#).

20.10 Frequently Asked Questions

20.10.1 Appliances

What is an appliance?

An appliance is an application packaged together with the environment that this application requires. For example, an appliance can consist of the operating system, libraries and tools used by the application, configuration features such as environment variable settings, and the installation of the application itself. Examples of appliances might include a KVM virtual machine image, a Docker image, or a bare metal image. Chameleon appliance refers to bare metal images that can be deployed on the Chameleon testbed. Since an appliance captures the experimental environment exactly, it is a key element of reproducibility; publishing an appliance used to obtain experimental results will go a long way to allowing others to reproduce and build on your research easily.

To deploy distributed applications on several Chameleon instances, complex appliances combine an image and a template describing how the cluster should be configured and contextualized. You can read more about them in the [Complex Appliances documentation](#).

What is the Appliance Catalog?

The [Chameleon Appliance Catalog](#) is a repository that allows users to discover, publish, and share appliances. The appliance catalog contains useful images of both bare metal and virtual machine appliances supported by the Chameleon team as well as appliances contributed by users.

How do I publish an appliance in the Appliance Catalog?

The new Appliance Catalog allows you to easily publish and share your own appliances so that others can discover them and use them either to reproduce the research of others or as a basis for their own research. Before creating your own appliance it is advisable to review other appliances on the [Chameleon Appliance Catalog](#) in order to get an idea of the categories you will want to contribute and what others have done.

Once you are ready to proceed, an appliance can be contributed to Chameleon in the following steps:

1. Create the appliance itself. You may want to test it as well as give some thought to what support you are willing to provide for the appliance (e.g., if your group developed and supports a software package, the appliance may be just a new way of packaging the software and making it available, in which case your standard support channels may be appropriate for the appliance as well).
2. Upload the appliance to the Chameleon Image Repository (Glance) and make the image public. In order to enter the appliance into the Catalog you will be asked to provide the Glance ID for the image. These IDs are per-cloud, so that there are three options right now: bare metal/CHI at University of Chicago, bare metal/CHI at TACC, and OpenStack/KVM at TACC. You will need to provide at least one appliance, but may want to provide all three.
3. Go to the [Appliance Catalog Create Appliance web form](#), fill out, and submit the form. Be prepared to provide the following information: a descriptive name (this sometimes requires some thought!), author and support contact, version, and an informative description. The description is a very important part of the appliance record; others will use it to evaluate if the appliance contains tools they need for their research so it makes sense to prepare it carefully. To make your description effective you may want to think of the following questions: what does the appliance contain? what are the specific packages and their versions? what is it useful for? where can it be deployed and/or what restrictions/limitations does it have? how should users connect to it / what accounts are enabled?

If you are adding a complex appliance, skip the image ID fields and enter your template instead in the dedicated text box.

As always, if you encounter any problems or want to share with us additional improvements we should do to the process, please don't hesitate to [submit a ticket](#).

How can I manage an appliance on Appliance Catalog?

If you are the owner of the appliance, you can edit the appliance data, such as the description or the support information. Browse to the appliance that you want to edit and view its Details page. At the top right of the page is an Edit button. You will be presented with the same web form as when creating the appliance, pre-filled with the appliances current information. Make changes as necessary and click Save at the bottom of the page.

And finally, you can delete appliances you had made available. Browse to the appliance that you want to delete and click Edit on the Appliance Details page. At the bottom of the page is a Delete button. You will be asked to confirm once more that you do want to delete this appliance. After confirming, the appliance will be removed and no longer listed on the Appliance Catalog.

Why are there different image IDs for the same appliance?

The three clouds forming the Chameleon testbed are fully separated, each having its own Glance image repository. The same appliance image uploaded to the three clouds will produce three different image IDs.

In addition, it is sometimes needed to customize an appliance image for each site, resulting in slightly different image files.

Can I use another operating system on bare-metal?

The recommended appliance for Chameleon is CentOS 7 (supported by Chameleon staff), or appliances built on top of it.

These appliances provide Chameleon-specific customizations, such as login using the cc account, the cc-checks utility to verify hardware against our resource registry, gathering of metrics, etc.

Since 2016, we also provide and support Ubuntu 14.04 and 16.04 appliances with the same functionality.

20.10.2 Bare Metal Troubleshooting

Why are my Bare Metal instances failing to launch?

The Chameleon Bare Metal clouds require users to reserve resources before allowing them to launch instances. Please follow the [documentation](#) and make sure that:

1. You have created a lease and it has started (the associated reservation is shown as **Active**)
2. You have selected your reservation in the **Launch Instance** panel

If you still cannot start instances, please [open a ticket with our help desk](#).

20.10.3 OpenStack KVM Troubleshooting

Why are my OpenStack KVM instances failing to launch?

If you get an error stating that **No valid host was found**, it might be caused by a lack of resources in the cloud. The Chameleon staff continuously monitors the utilization of the testbed, but there might be times when no more resources are available. If the error persists, please [open a ticket with our help desk](#).

Why can't I ping or SSH to my instance?

While the possibility that the system is being taken over by nanites should not be discounted too easily, it is always prudent to first check for the following issues:

- Do you have a floating IP associated with your instance? By default, instances do not have publicly-accessible IP addresses assigned. See the **Managing Virtual Machine Instances** section in the [User Guide](#).
- Does your security group allow incoming ICMP (e.g. ping) traffic? By default, firewall rules do not allow ping to your instances. If you wish to enable it, see the **Firewall (Access Security)** section in the [User Guide](#).
- Does your security group allow incoming SSH (TCP port 22) traffic? By default, firewall rules do not allow SSH to your instances. If you wish to enable it, see the **Firewall (Access Security)** section in the [User Guide](#).

If none of these solve your problem, please [open a ticket with our help desk](#), and send us the results of the above (and any evidence of nanites you find as well).



Python

21	Introduction	291
21.1	Introduction to Python	
21.2	References	
22	Install	293
22.1	Python Installation	
22.2	Interactive Python	
22.3	REPL (Read Eval Print Loop)	
22.4	Python 3 Features in Python 2	
23	Language	303
23.1	Statements and Strings	
23.2	Variables	
23.3	Data Types	
23.4	Module Management	
23.5	Date Time in Python	
23.6	Control Statements	
23.7	Datatypes	
23.8	Functions	
23.9	Classes	
23.10	Modules	
23.11	Lambda Expressions	
23.12	Generators	
23.13	Non Blocking Threads	
24	Data Management	317
24.1	Formats	
24.2	Encryption	
24.3	Database Access	
24.4	SQLite	
25	Libraries	323
25.1	Installing Libraries	
25.2	Using pip to Install Packages	
25.3	GUI	
25.4	Formatting and Checking Python Code	
25.5	Using autopep8	
25.6	Writing Python 3 Compatible Code	
25.7	Using Python on FutureSystems	
25.8	Ecosystem	
25.9	Resources	
25.10	Exercises	
25.11	Python for Big Data	
25.12	Parsing Data	



21. Introduction

21.1 Introduction to Python

Portions of this lesson have been adapted from the [official Python Tutorial](#) copyright [Python Software Foundation](#).

Python is an easy to learn programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's simple syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter can be extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python is an interpreted, dynamic, high-level programming language suitable for a wide range of applications.

The philosophy of python is summarized in [The Zen of Python](#) as follows:

- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

The main features of Python are:

- Use of indentation whitespace to indicate blocks

- Object orient paradigm
- Dynamic typing
- Interpreted runtime
- Garbage collected memory management
- a large standard library
- a large repository of third-party libraries

Python is used by many companies (such as Google, Yahoo!, CERN, NASA) and is applied for web development, scientific computing, embedded applications, artificial intelligence, software development, and information security, to name a few.

The material collected here introduces the reader to the basic concepts and features of the Python language and system. After you have worked through the material you will be able to:

- use Python
- use the interactive Python interface
- understand the basic syntax of Python
- write and run Python programs stored in a file
- have an overview of the standard library
- install Python libraries using `pyenv` or if it is not available `virtualenv`

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules.

In order to conduct this lesson you need

- A computer with Python 2.7.13 or 3.6.2
- Familiarity with command line usage
- A text editor such as [PyCharm](#), `emacs`, `vi` or others. You should identify which works best for you and set it up.

21.2 References

Some important additional information can be found on the following Web pages.

- [Python](#)
- [Pip](#)
- [Virtualenv](#)
- [NumPy](#)
- [SciPy](#)
- [Matplotlib](#)
- [Pandas](#)
- [pyenv](#)
- [PyCharm](#)

Python module of the week is a Web site that provides a number of short examples on how to use some elementary python modules. Not all modules are equally useful and you should decide if there are better alternatives. However for beginners this site provides a number of good examples

- Python 2: <https://pymotw.com/2/>
- Python 3: <https://pymotw.com/3/>



22. Install

22.1 Python Installation

Python is easy to install and very good instructions for most platforms can be found on the python.org Web page. We will be using Python 2.7.13 and/or Python 3 in our activities.

To manage python modules, it is useful to have [pip](https://pip.pypa.io/) package installation tool on your system.

In the tutorial, we assume that you have a computer with python installed. However, we also recommend that for the class you use Python's `virtualenv` (see below) to isolate your development Python from the system installed Python.

22.1.1 Managing custom Python installs

Often you have your own computer and you do not like to change its environment to keep it in pristine condition. Python comes with many libraries that could for example conflict with libraries that you have installed. To avoid this it is better to work in an isolated python we can use tools such as `virtualenv`, `pyenv` or `pyenv` for 3.6.4¹. Which you use depends on you, but we highly recommend `pyenv` if you can.

Managing Multiple Python Versions with Pyenv

Python has several versions that are used by the community. This includes Python 2 and Python 3, but all different management of the python libraries. As each OS may have their own version of python installed. It is not recommended that you modify that version. Instead you may want to create a localized python installation that you as a user can modify. To do that we

¹check for the newest version

recommend *pyenv*. Pyenv allows users to switch between multiple versions of Python (<https://github.com/yyuu/pyenv>). To summarize:

- users to change the global Python version on a per-user basis;
- users to enable support for per-project Python versions;
- easy version changes without complex environment variable management;
- to search installed commands across different python versions;
- integrate with tox (<https://tox.readthedocs.io/>).

Installation without pyenv

If you need to have more than one python version installed and do not want or can use pyenv, we recommend you download and install python 2.7.13 and 3.6.4² from python.org (<https://www.python.org/downloads/>)

Disabling wrong python installs on OSX

While working with students we have seen at times that they take other classes either at universities or online that teach them how to program in python. Unfortunately, although they seem to do that they often ignore to teach you how to properly install python. I just recently had a students that had installed python 7 times on his OSX machine, while another student had 3 different installations, all of which conflicted with each other as they were not set up properly.

We recommend that you inspect if you have a files such as `~/.bashrc` or `~/.bashrc_profile` in your home directory and identify if it activates various versions of python on your computer. If so you could try to deactivate them while out-commenting the various versions with the `#` character at the beginning of the line, start a new terminal and see if the terminal shell still works. Than you can follow our instructions here while using an install on pyenv.

Install pyenv on OSX from git

This is our recommended way to install pyenv on OSX:

```
$ git clone https://github.com/pyenv/pyenv.git ~/.pyenv
$ git clone https://github.com/pyenv/pyenv-virtualenv.git ~/.pyenv/plugins/pyenv-virtualenv
$ git clone https://github.com/yyuu/pyenv-virtualenvwrapper.git ~/.pyenv/plugins/pyenv-virtualenvwrap
$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bash_profile
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bash_profile
```

Installation of Homebrew

Before installing anything on your computer make sure you have enough space. Use in the terminal the command:

```
$ df -h
```

which gives your an overview of your file system. If you do not have enough space, please make sure you free up unused files from your drive.

In many occasions it is beneficial to use readline as it provides nice editing features for the terminal and xz for completion. First, make sure you have xcode installed:

```
$ xcode-select --install
```

Next install homebrew, pyenv, pyenv-virtualenv and pyenv-virtualwrapper. Additionally install

²check for the newest version

readline and some compression tools:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew update
brew install readline xz
```

Install pyenv on OSX with Homebrew

We describe here a mechanism of installing pyenv with homebrew. Other mechanisms can be found on the pyenv documentation page (<https://github.com/yyuu/pyenv-installer>). You must have homebrew installed as discussed in the previous section.

To install pyenv with homebrew execute in the terminal:

```
brew install pyenv pyenv-virtualenv pyenv-virtualenvwrapper
```

Install pyenv on Ubuntu

The following steps will install pyenv in a new ubuntu 16.04 distribution.

Start up a terminal and execute in the terminal the following commands. We recommend that you do it one command at a time so you can observe if the command succeeds:

```
$ sudo apt-get update
$ sudo apt-get install git python-pip make build-essential libssl-dev
$ sudo apt-get install zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev
$ sudo pip install virtualenvwrapper

$ git clone https://github.com/yyuu/pyenv.git ~/.pyenv
$ git clone https://github.com/pyenv/pyenv-virtualenv.git ~/.pyenv/plugins/pyenv-virtualenv
$ git clone https://github.com/yyuu/pyenv-virtualenvwrapper.git ~/.pyenv/plugins/pyenv-virtualenvwrap

$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
```

Now that you have installed pyenv it is not yet activated in your current terminal. The easiest thing to do is to start a new terminal and typ in:

```
which pyenv
```

If you see a response pyenv is installed and you can proceed with the next steps.

Please remember whenever you modify `.bashrc` or `.bash_profile` you need to start a new terminal.

Install Different Python Versions

Pyenv provides a large list of different python versions. To see the entire list please use the command:

```
$ pyenv install -l
```

However, for us we only need to worry about python 2.7.13 and python 3.6.4³. You can now install different versions of python into your local environment with the following commands:

```
$ pyenv install 2.7.13
$ pyenv install 3.6.4
```

You can set the global python default version with:

```
$ pyenv global 2.7.13
```

³check for the newest version

Type the following to determine which version you activated:

```
$ pyenv version
```

Type the following to determine which versions you have available:

```
$ pyenv versions
```

Associate a specific environment name with a certain python version, use the following commands:

```
$ pyenv virtualenv 2.7.13 ENV2
$ pyenv virtualenv 3.6.4 ENV3
```

In the example above, ENV2 would represent python 2.7.13 while ENV3 would represent python 3.6.4. Often it is easier to type the alias rather than the explicit version.

Set up the Shell

To make all work smoothly from your terminal, you can include the following in your `.bashrc` files:

```
export PYENV_VIRTUALENV_DISABLE_PROMPT=1
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"

__pyenv_version_ps1() {
  local ret=$?;
  output=$(pyenv version-name)
  if [[ ! -z $output ]]; then
    echo -n "($output)"
  fi
  return $ret;
}

PS1="\${__pyenv_version_ps1} ${PS1}"
```

We recommend that you do this towards the end of your file.

Switching Environments

After setting up the different environments, switching between them is now easy. Simply use the following commands:

```
(2.7.13) $ pyenv activate ENV2
(ENV2) $ pyenv activate ENV3
(ENV3) $ pyenv activate ENV2
(ENV2) $ pyenv deactivate ENV2
(2.7.13) $
```

To make it even easier, you can add the following lines to your `.bash_profile` file:

```
alias ENV2="pyenv activate ENV2"
alias ENV3="pyenv activate ENV3"
```

If you start a new terminal, you can switch between the different versions of python simply by typing:

```
$ ENV2
$ ENV3
```

22.1.2 Updating Python Version List

Pyenv maintains locally a list of available python versions. To see the list use the command

```
1 pyenv install -l
```

To obtain the newest list please use the command

```
1 cd ~/.pyenv/plugins/python-build/../../ && git pull
```

Now when you call

```
1 pyenv install -l
```

You will see the updated list.

22.1.3 Updating to a new version of Python with pyenv

Naturally python itself evolves and new versions will become available via pyenv. To facilitate such a new version you need to first install it into pyenv. Let us assume you had an old version of python installed onto the ENV3 environment. Then you need to execute the following steps:

```
1 pyenv deactivate
2 pyenv uninstall ENV3
3 pyenv install 3.6.4
4 pyenv virtualenv 3.6.4 ENV3
5 ENV3
6 pip install pip -U
7 \begin{lstlisting}
8
9 With the pi install command, we make sure we have the newest version
10 of pip. In case you get an error, you may have to update xcode as follows ↔
   ↔ and
11 try again:
12
13 \begin{lstlisting}
14 xcode-select --install
```

After you installed it you can activate it by typing ENV3. Naturally this requires that you added it to your bash environment as discussed in Section 22.1.1.

22.1.4 Installation without pyenv

If you need to have more than one python version installed and do not want or can use pyenv, we recommend you download and install python 2.7.13 and 3.6.4 from python.org (<https://www.python.org/downloads/>)

Make sure pip is up to date

As you will want to install other packages, make sure pip is up to date:

```
pip install pip -U
```

```
pyenv virtualenv anaconda3-4.3.1 ANA3 pyenv activate ANA3
```

22.1.5 Anaconda and Miniconda

We do not recommend that you use anaconda or miniconda as it may interfere with your default python interpreters and setup.

Please note that beginners to python should always use anaconda or miniconda only after they have installed pyenv and use it. For this class neither anaconda nor miniconda is required. In fact we do not recommend it. We keep this section as we know that other classes at IU may use anaconda. We are not aware if these classes teach you the right way to install it, with *pyenv*.

Miniconda

This section about miniconda is experimental and has not been tested. We are looking for contributors that help completing it. If you use anaconda or miniconda we recommend to manage it via *pyenv*.

To install mini conda you can use the following commands:

```
$ mkdir ana
$ cd ana
$ pyenv install miniconda3-latest
$ pyenv local miniconda3-latest
$ pyenv activate miniconda3-latest
$ conda create -n ana anaconda
```

To activate use:

```
$ source activate ana
```

To deactivate use:

```
$ source deactivate
```

To install cloudmesh cmd5 please use:

```
$ pip install cloudmesh.cmd5
$ pip install cloudmesh.sys
```

Anaconda

This section about anaconda is experimental and has not been tested. We are looking for contributors that help completing it.

You can add anaconda to your *pyenv* with the following commands:

```
pyenv install anaconda3-4.3.1
```

To switch more easily we recommend that you use the following in your *.bash_profile* file:

```
alias ANA="pyenv activate anaconda3-4.3.1"
```

Once you have done this you can easily switch to anaconda with the command:

```
$ ANA
```

Terminology in anaconda could lead to confusion. Thus we like to point out that the version number of anaconda is unrelated to the python version. Furthermore, anaconda uses the term *root* not for the root user, but for the originating directory in which the anaconda program is installed.

In case you like to build your own conda packages at a later time we recommend that you install the *conda-build* package:

```
$ conda install conda-build
```


When executing:

```
pyenv versions
```

you will see after the install completed the anaconda versions installed:

```
pyenv versions
system
2.7.13
2.7.13/envs/ENV2
3.6.4
3.6.4/envs/ENV3
ENV2
ENV3
* anaconda3-4.3.1 (set by PYENV_VERSION environment variable)
```

Let us now create virtualenv for anaconda:

```
$ pyenv virtualenv anaconda3-4.3.1 ANA
```

To activate it you can now use:

```
$ pyenv ANA
```

However, anaconda may modify your `.bashrc` or `.bash_profile` files and may result in incompatibilities with other python versions. For this reason we recommend not to use it. If you find ways to get it to work reliably with other versions, please let us know and we update this tutorial.

To install cloudmesh cmd5 please use:

```
$ pip install cloudmesh.cmd5
$ pip install cloudmesh.sys
```

Exercise

Exercise 22.1 Write installation instructions for an operating system of your choice and add to this documentation. ■

Exercise 22.2 Replicate the steps above, so you can type in ENV2 and ENV3 in your terminals to switch between python 2 and 3. ■

virtualenv

environment while using virtualenv,. Documentation about it can be found at:

```
* https://virtualenv.pypa.io
```

The installation is simple once you have pip installed. If it is not installed you can say:

```
$ easy_install pip
```

After that you can install the virtual env with:

```
$ pip install virtualenv
```

To setup an isolated environment for example in the directory `~/ENV` please use:

```
$ virtualenv ~/ENV
```

To activate it you can use the command:

```
$ source ~/ENV/bin/activate
```

you can put this command in your `.bashrc` or `.bash_profile` files so you do not forget to activate it. Instructions for this can be found in our lesson on Linux `bashrc`.

22.2 Interactive Python

Python can be used interactively. You can enter the interactive mode by entering the interactive loop by executing the command:

```
1 $ python
```

You will see something like the following:

```
1 Python 2.7.13 (default, Nov 19 2016, 06:48:10)
2 [GCC 5.4.0 20160609] on linux2
3 Type "help", "copyright", "credits" or "license" for more information.
4 >>>
```

The >>> is the prompt used by the interpreter. This is similar to bash where commonly \$ is used.

Sometimes it is convenient to show the prompt when illustrating an example. This is to provide some context for what we are doing. If you are following along you will not need to type in the prompt.

This interactive python process does the following:

- *read* your input commands
- *evaluate* your command
- *print* the result of evaluation
- *loop* back to the beginning.

This is why you may see the interactive loop referred to as a **REPL**: **Read-Evaluate-Print-Loop**.

22.3 REPL (Read Eval Print Loop)

There are many different types beyond what we have seen so far, such as **dictionaries**, **lists**, **sets**. One handy way of using the interactive python is to get the type of a value using `type()`:

```
1 >>> type(42)
2 <type 'int'>
3 >>> type(hello)
4 <type 'str'>
5 >>> type(3.14)
6 <type 'float'>
```

You can also ask for help about something using `help()`:

```
1 >>> help(int)
2 >>> help(list)
3 >>> help(str)
```

Using `help()` opens up a help message within a pager. To navigate you can use the spacebar to go down a page w to go up a page, the arrow keys to go up/down line-by-line, or q to exit.

22.4 Python 3 Features in Python 2

In this course we want to be able to seamlessly switch between python 2 and python 3. Thus it is convenient from the start to use python 3 syntax when it is supported also in python 2. One of the most used functions is the print statement that has in python 3 parentheses. To enable it in python 2 you just need to import this function:

```
1 >>> from __future__ import print_function, division
```

The first of these imports allows us to use the print function to output text to the screen, instead of the print statement, which Python 2 uses. This is simply a [design decision](#) that better reflects Python's underlying philosophy.

Other functions such as the division also behave differently. Thus we use

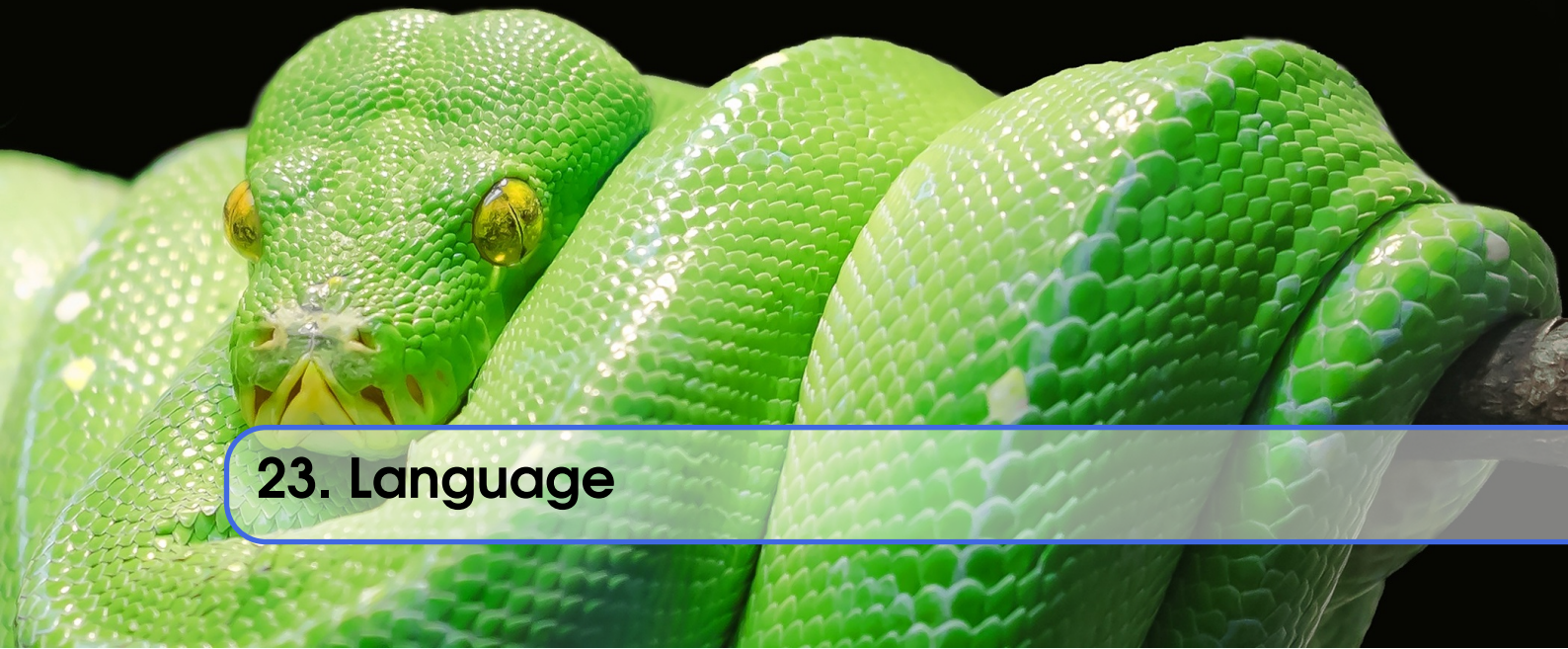
```
1 >>> from __future__ import division
```

This import makes sure that the [division operator](#) behaves in a way a newcomer to the language might find more intuitive. In Python 2, division / is *floor division* when the arguments are integers, meaning that the following

```
1 (5 / 2 == 2) is True
```

In Python 3, division / is a floating point division, thus

```
1 (5 / 2 == 2.5) is True
```

23. Language

TODO: TA: some of the python examples assume REPL, but its better to use a print statement instead as more general, please fix

23.1 Statements and Strings

Let us explore the syntax of Python. Type into the interactive loop and press Enter:

```
1 print("Hello world from Python!")
2 # Hello world from Python!
```

What happened: the print function was given a **string** to process. A string is a sequence of characters. A **character** can be a alphabetic (A through Z, lower and upper case), numeric (any of the digits), white space (spaces, tabs, newlines, etc), syntactic directives (comma, colon, quotation, exclamation, etc), and so forth. A string is just a sequence of the character and typically indicated by surrounding the characters in double quotes.

Standard output is discussed in the `../..../lesson/linux/shell` lesson.

So, what happened when you pressed Enter? The interactive Python program read the line `print ("Hello world from Python!")`, split it into the print statement and the "Hello world from Python!" string, and then executed the line, showing you the output.

23.2 Variables

You can store data into a **variable** to access it later. For instance, instead of:

```
3 print('Hello world from Python!')
```

which is a lot to type if you need to do it multiple times, you can store the string in a variable for convenient access:

```
4 hello = 'Hello world from Python!'
5 print(hello)
6 # Hello world from Python!
```

23.3 Data Types

23.3.1 Booleans

A **boolean** is a value that indicates *truthness* of something. You can think of it as a toggle: either “on” or “off”, “one” or “zero”, “true” or “false”. In fact, the only possible values of the **boolean** (or `bool`) type in Python are:

- `True`
- `False`

You can combine booleans with **boolean operators**:

- `and`
- `or`

```
7 print(True and True)
8 # True
9
10 print(True and False)
11 # False
12
13 print(False and False)
14 # False
15
16 print(True or True)
17 # True
18
19 print(True or False)
20 # True
21
22 print(False or False)
23 # False
```

23.3.2 Numbers

The interactive interpreter can also be used as a calculator. For instance, say we wanted to compute a multiple of 21:

```
24 print(21 * 2)
25 # 42
```

We saw here the print statement again. We passed in the result of the operation `21 * 2`. An **integer** (or **int**) in Python is a numeric value without a fractional component (those are called **floating point** numbers, or **float** for short).

The mathematical operators compute the related mathematical operation to the provided numbers. Some operators are:

```
* --- multiplication
/ --- division
+ --- addition
- --- subtraction
** --- exponent
```

Exponentiation x^y is written as `x**y` is x to the yth power.

You can combine **floats** and **ints**:

```
26 print(3.14 * 42 / 11 + 4 - 2)
27 # 13.9890909091
28
29 print(2**3)
30 # 8
```

Note that **operator precedence** is important. Using parenthesis to indicate affect the order of operations gives a difference results, as expected:

```
31 print(3.14 * (42 / 11) + 4 - 2)
32 # 11.42
33
34 print(1 + 2 * 3 - 4 / 5.0)
35 # 6.2
36
37 print( (1 + 2) * (3 - 4) / 5.0 )
38 # -0.6
```

23.4 Module Management

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. A module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

23.4.1 Import Statement

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. The `from...import` Statement Python's `from` statement lets you import specific attributes from a module into the current namespace. It is preferred to use for each import its own line such as:

```
39 import numpy
40 import matplotlib
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module.

23.4.2 The `from ... import` Statement

Python's `from` statement lets you import specific attributes from a module into the current namespace. The `from ... import` has the following syntax:

```
41 from datetime import datetime
```

23.5 Date Time in Python

The datetime module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation. For related functionality, see also the time and calendar modules.

The import Statement You can use any Python source file as a module by executing an import statement in some other Python source file.

```
42 from datetime import datetime
```

This module offers a generic date/time string parser which is able to parse most known formats to represent a date and/or time.

```
43 from dateutil.parser import parse
```

pandas is an open source Python library for data analysis that needs to be imported.

```
44 import pandas as pd
```

Create a string variable with the class start time

```
45 fall_start = '08-21-2017'
```

Convert the string to datetime format

```
46 datetime.strptime(fall_start, '%m-%d-%Y')
47 # datetime.datetime(2017, 8, 21, 0, 0)
```

Creating a list of strings as dates

```
48 class_dates = ['8/25/2017', '9/1/2017', '9/8/2017', '9/15/2017', '9/22/2017' ←
↪ ', '9/29/2017']
```

Convert Class_dates strings into datetime format and save the list into variable a

```
49 a = [datetime.strptime(x, '%m/%d/%Y') for x in class_dates]
```

Use parse() to attempt to auto-convert common string formats. Parser must be a string or character stream, not list.

```
50 parse(fall_start)
51 # datetime.datetime(2017, 8, 21, 0, 0)
```

Use parse() on every element of the Class_dates string.

```
52 [parse(x) for x in class_dates]
53 # [datetime.datetime(2017, 8, 25, 0, 0),
54 #  datetime.datetime(2017, 9, 1, 0, 0),
55 #  datetime.datetime(2017, 9, 8, 0, 0),
56 #  datetime.datetime(2017, 9, 15, 0, 0),
57 #  datetime.datetime(2017, 9, 22, 0, 0),
58 #  datetime.datetime(2017, 9, 29, 0, 0)]
```

Use parse, but designate that the day is first.


```

59 parse (fall_start, dayfirst=True)
60 # datetime.datetime(2017, 8, 21, 0, 0)

```

Create a dataframe. A DataFrame is a tabular data structure comprised of rows and columns, akin to a spreadsheet, database table. DataFrame as a group of Series objects that share an index (the column names).

```

61 import pandas as pd
62 data = {'class_dates': ['8/25/2017 18:47:05.069722',
63                        '9/1/2017 18:47:05.119994',
64                        '9/8/2017 18:47:05.178768',
65                        '9/15/2017 18:47:05.230071',
66                        '9/22/2017 18:47:05.230071',
67                        '9/29/2017 18:47:05.280592'],
68        'complete': [1, 0, 1, 1, 0, 1]}
69 df = pd.DataFrame(data, columns = ['class_dates', 'complete'])
70 print(df)
71 #                class_dates  complete
72 #  0  8/25/2017 18:47:05.069722         1
73 #  1   9/1/2017 18:47:05.119994         0
74 #  2   9/8/2017 18:47:05.178768         1
75 #  3  9/15/2017 18:47:05.230071         1
76 #  4  9/22/2017 18:47:05.230071         0
77 #  5  9/29/2017 18:47:05.280592         1

```

Convert df['date'] from string to datetime

```

78 import pandas as pd
79 pd.to_datetime(df['class_dates'])
80 # 0   2017-08-25 18:47:05.069722
81 # 1   2017-09-01 18:47:05.119994
82 # 2   2017-09-08 18:47:05.178768
83 # 3   2017-09-15 18:47:05.230071
84 # 4   2017-09-22 18:47:05.230071
85 # 5   2017-09-29 18:47:05.280592
86 # Name: class_dates, dtype: datetime64[ns]

```

23.6 Control Statements

23.6.1 Comparison

Computer programs do not only execute instructions. Occasionally, a choice needs to be made. Such as a choice is based on a condition. Python has several conditional operators:

> greater than
 < smaller than
 == equals
 != is not

Conditions are always combined with variables. A program can make a choice using the if keyword. For example:

```

87 x = int(input("Guess x:"))
88 if x == 4:
89     print('You guessed correctly!')

```

In this example, *You guessed correctly!* will only be printed if the variable `x` equals to four (see table above). Python can also execute multiple conditions using the `elif` and `else` keywords.

```

90 x = int(input("Guess x:"))
91 if x == 4:
92     print('You guessed correctly!')
93 elif abs(4 - x) == 1:
94     print('Wrong guess, but you are close!')
95 else:
96     print('Wrong guess')
```

23.6.2 Iteration

To repeat code, the `for` keyword can be used. For example, to display the numbers from 1 to 10, we could write something like this:

```

97 for i in range(1, 11):
98     print('Hello!')
```

The second argument to `range`, *11*, is not inclusive, meaning that the loop will only get to *10* before it finishes. Python itself starts counting from 0, so this code will also work:

```

99 for i in range(0, 10):
100     print(i + 1)
```

In fact, the `range` function defaults to starting value of 0, so the above is equivalent to:

```

101 for i in range(10):
102     print(i + 1)
```

We can also nest loops inside each other:

```

103 for i in range(0,10):
104     for j in range(0,10):
105         print(i, ' ',j)
```

In this case we have two nested loops. The code will iterate over the entire coordinate range (0,0) to (9,9)

23.7 Datatypes

23.7.1 Lists

see: https://www.tutorialspoint.com/python/python_lists.htm

Lists in Python are ordered sequences of elements, where each element can be accessed using a 0-based index.

To define a list, you simply list its elements between square brackets '[']:

```

106 names = ['Albert', 'Jane', 'Liz', 'John', 'Abby']
107 names[0] # access the first element of the list
108 # 'Albert'
109 names[2] # access the third element of the list
110 # 'Liz'
```

You can also use a negative index if you want to start counting elements from the end of the list. Thus, the last element has index `-1`, the second before last element has index `-2` and so on:

```
111 names[-1] # access the last element of the list
112 # 'Abby'
113 names[-2] # access the second last element of the list
114 # 'John'
```

Python also allows you to take whole slices of the list by specifying a beginning and end of the slice separated by a colon

```
115 names[1:-1] # the middle elements, excluding first and last
116 # ['Jane', 'Liz', 'John']
```

As you can see from the example above, the starting index in the slice is inclusive and the ending one, exclusive.

Python provides a variety of methods for manipulating the members of a list.

You can add elements with `append`:

```
117 names.append('Liz')
118 names
119 # ['Albert', 'Jane', 'Liz', 'John', 'Abby', 'Liz']
```

As you can see, the elements in a list need not be unique.

Merge two lists with `extend`:

```
120 names.extend(['Lindsay', 'Connor'])
121 names
122 # ['Albert', 'Jane', 'Liz', 'John', 'Abby', 'Liz', 'Lindsay', 'Connor']
```

Find the index of the first occurrence of an element with `index`:

```
123 names.index('Liz')
124 # 2
```

Remove elements by value with `remove`:

```
125 names.remove('Abby')
126 names
127 # ['Albert', 'Jane', 'Liz', 'John', 'Liz', 'Lindsay', 'Connor']
```

Remove elements by index with `pop`:

```
128 names.pop(1)
129 # 'Jane'
130 names
131 # ['Albert', 'Liz', 'John', 'Liz', 'Lindsay', 'Connor']
```

Notice that `pop` returns the element being removed, while `remove` does not.

If you are familiar with stacks from other programming languages, you can use `insert` and `pop`:

```
132 names.insert(0, 'Lincoln')
133 names
134 # ['Lincoln', 'Albert', 'Liz', 'John', 'Liz', 'Lindsay', 'Connor']
135 names.pop()
136 # 'Connor'
137 names
138 # ['Lincoln', 'Albert', 'Liz', 'John', 'Liz', 'Lindsay']
```

The Python documentation contains a full list of list operations.

To go back to the range function you used earlier, it simply creates a list of numbers:

```
139 range(10)
140 # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
141 range(2, 10, 2)
142 # [2, 4, 6, 8]
```

23.7.2 Sets

Python lists can contain duplicates as you saw above:

```
143 names = ['Albert', 'Jane', 'Liz', 'John', 'Abby', 'Liz']
```

When we don't want this to be the case, we can use a *set*:

```
144 unique_names = set(names)
145 unique_names
146 # set(['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay'])
```

Keep in mind that the *set* is an unordered collection of objects, thus we can not access them by index:

```
147 unique_names[0]
148 # Traceback (most recent call last):
149 #   File "<stdin>", line 1, in <module>
150 #   TypeError: 'set' object does not support indexing
```

However, we can convert a set to a list easily:

```
151 unique_names = list(unique_names)
152 unique_names ['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay']
153 unique_names[0]
154 # 'Lincoln'
```

Notice that in this case, the order of elements in the new list matches the order in which the elements were displayed when we create the set. We had

```
set(['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay'])
```

and now we have

```
['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay'])
```

You should not assume this is the case in general. That is, don't make any assumptions about the order of elements in a set when it is converted to any type of sequential data structure.

You can change a set's contents using the add, remove and update methods which correspond to the append, remove and extend methods in a list. In addition to these, *set* objects support the operations you may be familiar with from mathematical sets: *union*, *intersection*, *difference*, as well as operations to check containment. You can read about this in the [Python documentation for sets](#).

23.7.3 Removal and Testing for Membership in Sets

One important advantage of a set over a list is that **access to elements is fast**. If you are familiar with different data structures from a Computer Science class, the Python list is implemented by an array, while the set is implemented by a hash table.

We will demonstrate this with an example. Let's say we have a list and a set of the same number of elements (approximately 100 thousand):

```
155 import sys, random, timeit
156 nums_set = set([random.randint(0, sys.maxint) for _ in range(10**5)])
157 nums_list = list(nums_set)
158 len(nums_set)
159 # 100000
```

We will use the `timeit` Python module to time 100 operations that test for the existence of a member in either the list or set:

```
160 timeit.timeit('random.randint(0, sys.maxint) in nums',
161               setup='import random; nums=%s' % str(nums_set), number=100)
162 # 0.0004038810729980469
163 timeit.timeit('random.randint(0, sys.maxint) in nums',
164               setup='import random; nums=%s' % str(nums_list), number=100)
165 # 0.3980541229248047
```

The exact duration of the operations on your system will be different, but the take away will be the same: searching for an element in a set is orders of magnitude faster than in a list. This is important to keep in mind when you work with large amounts of data.

23.7.4 Dictionaries

One of the very important data structures in python is a dictionary also referred to as *dict*.

A dictionary represents a key value store:

```
166 person = {'Name': 'Albert', 'Age': 100, 'Class': 'Scientist'}
167 print("person['Name']: ", person['Name'])
168 # person['Name']: Albert
169 print("person['Age']: ", person['Age'])
170 # person['Age']: 100
```

You can delete elements with the following commands:

```
171 del person['Name'] # remove entry with key 'Name'
172 person
173 # {'Age': 100, 'Class': 'Scientist'}
174 person.clear() # remove all entries in dict
175 # person
176 # {}
177 del person # delete entire dictionary
178 person
179 # Traceback (most recent call last):
180 # File "<stdin>", line 1, in <module>
181 # NameError: name 'person' is not defined
```

You can iterate over a dict:

```
182 person = {'Name': 'Albert', 'Age': 100, 'Class': 'Scientist'}
183 for item in person:
184     print(item, person[item])
185
186 # Age 100
187 # Name Albert
188 # Class Scientist
```

23.7.5 Dictionary Keys and Values

You can retrieve both the keys and values of a dictionary using the `keys()` and `values()` methods of the dictionary, respectively:

```
189 person.keys()
190 # ['Age', 'Name', 'Class']
191 person.values()
192 # [100, 'Albert', 'Scientist']
```

Both methods return lists. Notice, however, that the order in which the elements appear in the returned lists (Age, Name, Class) is different from the order in which we listed the elements when we declared the dictionary initially (Name, Age, Class). It is important to keep this in mind: **you can't make any assumptions about the order in which the elements of a dictionary will be returned by the `keys()` and `values()` methods.**

However, you can assume that if you call `keys()` and `values()` in sequence, the order of elements will at least correspond in both methods. In the above example Age corresponds to 100, Name to Albert, and Class to Scientist, and you will observe the same correspondence in general as long as *keys() and values() are called one right after the other.*

23.7.6 Counting with Dictionaries

One application of dictionaries that frequently comes up is counting the elements in a sequence. For example, say we have a sequence of coin flips:

```
193 import random
194 die_rolls = [random.choice(['heads', 'tails']) for _ in range(10)]
195 # die_rolls
196 # ['heads', 'tails', 'heads', 'tails', 'heads', 'heads',
197    'tails', 'heads', 'heads']
```

The actual list `die_rolls` will likely be different when you execute this on your computer since the outcomes of the die rolls are random.

To compute the probabilities of heads and tails, we could count how many heads and tails we have in the list:

```
198 counts = {'heads': 0, 'tails': 0}
199 for outcome in coin_flips:
200     assert outcome in counts
201     counts[outcome] += 1
202 print('Probability of heads: %.2f' % (counts['heads'] / len(coin_flips)))
203 # Probability of heads: 0.70
204
205 print('Probability of tails: %.2f' % (counts['tails'] / sum(counts.values()) ←
    ↪ ))
206 # Probability of tails: 0.30
```

In addition to how we use the dictionary counts to count the elements of `coin_flips`, notice a couple things about this example:

1. We used the `assert outcome in counts` statement. The `assert` statement in Python allows you to easily insert debugging statements in your code to help you discover errors more quickly. `assert` statements are executed whenever the internal Python `__debug__` variable is set to

True, which is always the case unless you start Python with the `-O` option which allows you to run *optimized* Python.

2. When we computed the probability of tails, we used the built-in sum function, which allowed us to quickly find the total number of coin flips. sum is one of many built-in function you can [read about here](#).

23.8 Functions

You can reuse code by putting it inside a function that you can call in other parts of your programs. Functions are also a good way of grouping code that logically belongs together in one coherent whole. A function has a unique name in the program. Once you call a function, it will execute its body which consists of one or more lines of code:

```
207 def check_triangle(a, b, c):
208     return \
209         a < b + c and a > abs(b - c) and \
210         b < a + c and b > abs(a - c) and \
211         c < a + b and c > abs(a - b)
212
213 print(check_triangle(4, 5, 6))
```

The `def` keyword tells Python we are defining a function. As part of the definition, we have the function name, `check_triangle`, and the parameters of the function -- variables that will be populated when the function is called.

We call the function with arguments 4, 5 and 6, which are passed in order into the parameters `a`, `b` and `c`. A function can be called several times with varying parameters. There is no limit to the number of function calls.

It is also possible to store the output of a function in a variable, so it can be reused.

```
214 def check_triangle(a, b, c):
215     return \
216         a < b + c and a > abs(b - c) and \
217         b < a + c and b > abs(a - c) and \
218         c < a + b and c > abs(a - b)
219
220 result = check_triangle(4, 5, 6)
221 print(result)
```

23.9 Classes

A class is an encapsulation of data and the processes that work on them. The data is represented in member variables, and the processes are defined in the methods of the class (methods are functions inside the class). For example, let's see how to define a Triangle class:

```
222 class Triangle(object):
223
224     def __init__(self, length, width, height, angle1, angle2, angle3):
225         if not self._sides_ok(length, width, height):
226             print('The sides of the triangle are invalid.')
227         elif not self._angles_ok(angle1, angle2, angle3):
228             print('The angles of the triangle are invalid.')
```

```

229         self._length = length
230         self._width = width
231         self._height = height
232
233         self._angle1 = angle1
234         self._angle2 = angle2
235         self._angle3 = angle3
236
237
238     def _sides_ok(self, a, b, c):
239         return \
240             a < b + c and a > abs(b - c) and \
241             b < a + c and b > abs(a - c) and \
242             c < a + b and c > abs(a - b)
243
244     def _angles_ok(self, a, b, c):
245         return a + b + c == 180
246
247 triangle = Triangle(4, 5, 6, 35, 65, 80)

```

Python has full Object-oriented programming (OOP) capabilities, however we can not cover all of them in a quick tutorial, so please refer to the [Python docs on classes and OOP](#).

23.10 Modules

Now write this simple program and save it:

```

1 from __future__ import print_statement, division
2 print("Hello world!")

```

As a check, make sure the file contains the expected contents on the command line:

```

1 $ cat hello.py
2 from __future__ import print_statement, division
3 print("Hello world!")

```

To execute your program pass the file as a parameter to the python command:

```

1 $ python hello.py
2 Hello world!

```

Files in which Python code is stored are called **modules**. You can execute a Python module from the command line like you just did, or you can import it in other Python code using the import statement.

Let's write a more involved Python program that will receive as input the lengths of the three sides of a triangle, and will output whether they define a valid triangle. A triangle is valid if the length of each side is less than the sum of the lengths of the other two sides and greater than the difference of the lengths of the other two sides.:

```

1 """Usage: check_triangle.py [-h] LENGTH WIDTH HEIGHT
2
3 Check if a triangle is valid.
4
5 Arguments:
6   LENGTH   The length of the triangle.

```



```

7     WIDTH      The width of the traingle.
8     HEIGHT    The height of the triangle.
9
10    Options:
11    -h --help
12    """
13    from __future__ import print_function, division
14    from docopt import docopt
15
16    if __name__ == '__main__':
17        args = docopt(__doc__)
18        a, b, c = int(args['LENGTH']), int(args['WIDTH']), int(args['HEIGHT'])
19        valid_triangle = \
20            a < b + c and a > abs(b - c) and \
21            b < a + c and b > abs(a - c) and \
22            c < a + b and c > abs(a - b)
23        print('Triangle with sides %d, %d and %d is valid: %r' % (
24            a, b, c, valid_triangle
25        ))

```

Assuming we save the program in a file called `check_triangle.py`, we can run it like so:

```

1 $ python check_triangle.py 4 5 6
2 Triangle with sides 4, 5 and 6 is valid: True

```

Let us break this down a bit.

1. We are importing the `print_function` and `division` modules from Python 3 like we did earlier in this tutorial. It's a good idea to always include these in your programs.
2. We've defined a boolean expression that tells us if the sides that were input define a valid triangle. The result of the expression is stored in the `valid_triangle` variable. Inside are `True`, and `False` otherwise.
3. We've used the backslash symbol `\` to format our code nicely. The backslash simply indicates that the current line is being continued on the next line.
4. When we run the program, we do the check `if __name__ == '__main__'`. `__name__` is an internal Python variable that allows us to tell whether the current file is being run from the command line (value `__name__`), or is being imported by a module (the value will be the name of the module). Thus, with this statement we're just making sure the program is being run by the command line.
5. We are using the `docopt` module to handle command line arguments. The advantage of using this module is that it generates a usage help statement for the program and enforces command line arguments automatically. All of this is done by parsing the docstring at the top of the file.
6. In the print function, we are using [Python's string formatting capabilities](#) to insert values into the string we are displaying.

23.11 Lambda Expressions

TODO: contribute

23.12 Generators

TODO: contribute

23.13 Non Blocking Threads

TODO: contribute



24. Data Management

Obviously when dealing with big data we may not only be dealing with data in one format but in many different formats. It is important that you will be able to master such formats and seamlessly integrate in your analysis. Thus we provide some simple examples on which different data formats exist and how to use them.

24.1 Formats

24.1.1 Pickle

Python pickle allows you to save data in a python native format into a file that can later be read in by other programs. However, the data format may not be portable among different python versions thus the format is often not suitable to store information. Instead we recommend for standard data to use either json or yaml.

```
import pickle

flavor = { "small": 100,
           "medium": 1000,
           "large": 10000 }

pickle.dump( flavor, open( "data.p", "wb" ) )
```

To read it back in use

```
flavor = pickle.load( open( "data.p", "rb" ) )
```

24.1.2 Text Files

To read text files into a variable called content you can use

```
content = open('filename.txt', 'r').read()
```

You can also use the following code while using the convenient `with` statement

```
with open('filename.txt','r') as file:
    content = file.read()
```

To split up the lines of the file into an array you can do

```
with open('filename.txt','r') as file:
    lines = file.read().splitlines()
```

This can also be done with the built-in `readlines` function

```
lines = open('filename.txt','r').readlines()
```

In case the file is too big you will want to read the file line by line:

```
with open('filename.txt','r') as file:
    line = file.readline()
    print (line)
```

24.1.3 CSV Files

Often data is contained in comma separated values (CSV) within a file. To read such files you can use the `csv` package.

```
import csv
with open('data.csv', 'rb') as f:
    contents = csv.reader(f)
    for row in content:
        print row
```

Using `pandas` you can read them as follows.

```
import pandas as pd
df = pd.read_csv("example.csv")
```

There are many other modules and libraries that include CSV read functions. In case you need to split a single line by comma, you may also use the `split` function. However, remember it will split at every comma, including those contained in quotes. So this method although looking originally convenient has limitations.

24.1.4 Excel spread sheets

`Pandas` contains a method to read Excel files

```
import pandas as pd
filename = 'data.xlsx'
data = pd.ExcelFile(file)
df = data.parse('Sheet1')
```

24.1.5 YAML

YAML is a very important format as it allows you easily to structure data in hierarchical files. It is frequently used to coordinate programs while using `yaml` as the specification for configuration files, but also data files. To read in a `yaml` file the following code can be used

```
import yaml
with open('data.yaml', 'r') as f:
    content = yaml.load(f)
```

The nice part is that this code can also be used to verify if a file is valid yaml. To write data out we can use

```
with open('data.yml', 'w') as f:
    yaml.dump(data, f, default_flow_style=False)
```

The flow style set to false formats the data in a nice readable fashion with indentations.

24.1.6 JSON

```
import json
with open('strings.json') as f:
    content = json.load(f)
```

24.1.7 XML

TODO: Tutorial: Please contribute a XML python tutorial.

24.1.8 RDF

To read RDF files you will need to install RDFlib with

```
pip install rdflib
```

This will then allow you to read RDF files

```
from rdflib.graph import Graph
g = Graph()
g.parse("filename.rdf", format="format")
for entry in g:
    print(entry)
```

Good examples on using RDF are provided on the RDFlib Web page at <https://github.com/RDFLib/rdflib>

From the Web page we showcase also how to directly process RDF data from the Web

```
import rdflib
g=rdflib.Graph()
g.load('http://dbpedia.org/resource/Semantic_Web')

for s,p,o in g:
    print s,p,o
```

24.1.9 PDF

The Portable Document Format (PDF) has been made available by Adobe Inc. royalty free. This has enabled PDF to become a world wide adopted format that also has been standardized in 2008 (ISO/IEC 32000-1:2008, <https://www.iso.org/standard/51502.html>). A lot of research is published in papers making PDF one of the de-facto standards for publishing. However, PDF is difficult to parse and is focused on high quality output instead of data representation. Nevertheless, tools to manipulate PDF exist:

PDFMiner <https://pypi.python.org/pypi/pdfminer/> allows the simple translation of PDF into text that than can be further mined. The manual page helps to demonstrate some examples <http://euske.github.io/pdfminer/index.html>.

pdf-parser.py <https://blog.didierstevens.com/programs/pdf-tools/> parses pdf documents and identifies some structural elements that can then be further processed.

If you know about other tools, let us know.

24.1.10 HTML

A very powerful library to parse HTML Web pages is provided with <https://www.crummy.com/software/BeautifulSoup/>

More details about it are provided in the documentation page <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

TODO: Students: beautiful soup contribute tutorial

24.1.11 ConfigParser

- <https://pymotw.com/2/ConfigParser/>

24.1.12 ConfigDict

- <https://github.com/cloudmesh/cloudmesh.common/blob/master/cloudmesh/common/ConfigDict.py>

24.2 Encryption

Often we need to protect the information stored in a file. This is achieved with encryption. There are many methods of supporting encryption and even if a file is encrypted it may be target to attacks. Thus it is not only important to encrypt data that you do not want others to see but also to make sure that the system on which the data is hosted is secure. This is especially important if we talk about big data having a potential large effect if it gets into the wrong hands.

To illustrate one type of encryption that is non trivial we have chosen to demonstrate how to encrypt a file with an ssh key. In case you have openssl installed on your system, this can be achieved as follows.

```
#!/bin/sh

# Step 1. Creating a file with data
echo "Big Data is the future." > file.txt

# Step 2. Create the pem
openssl rsa -in ~/.ssh/id_rsa -pubout > ~/.ssh/id_rsa.pub.pem

# Step 3. look at the pem file to illustrate how it looks like (optional)
cat ~/.ssh/id_rsa.pub.pem

# Step 4. encrypt the file into secret.txt
openssl rsautl -encrypt -pubin -inkey ~/.ssh/id_rsa.pub.pem -in file.txt -out secret.txt

# Step 5. decrypt the file and print the contents to stdout
openssl rsautl -decrypt -inkey ~/.ssh/id_rsa -in secret.txt
```

Most important here are Step 4 that encrypts the file and Step 5 that decrypts the file. Using the Python os module it is straight forward to implement this. However, we are providing in cloudmesh

a convenient class that makes the use in python very simple.

```
from cloudmesh.common.ssh.encrypt import EncryptFile

e = EncryptFile('file.txt', 'secret.txt')
e.encrypt()
e.decrypt()
```

In our class we initialize it with the locations of the file that is to be encrypted and decrypted. To initiate that action just call the methods `encrypt` and `decrypt`.

24.3 Database Access

TODO: Students: define conventional database access tutorial

see: https://www.tutorialspoint.com/python/python_database_access.htm

24.4 SQLite

TODO: Students: defineSQLite database access tutorial

<https://www.sqlite.org/index.html>

<https://docs.python.org/3/library/sqlite3.html>

24.4.1 Exercises

- Exercise 24.1** Test out the shell script to replicate how this example works ■
- Exercise 24.2** Test out the cloudmesh encryption class ■
- Exercise 24.3** What other encryption methods exist. Can you provide an example and contribute to the section? ■
- Exercise 24.4** What is the issue of encryption that make it challenging for Big Data ■
- Exercise 24.5** Given a test dataset with many files text files, how long will it take to encrypt and decrypt them on various machines. Write a benchmark that you test. Develop this benchmark as a group, test out the time it takes to execute it on a variety of platforms. ■



25. Libraries

25.1 Installing Libraries

Often you may need functionality that is not present in Python's standard library. In this case you have two options:

- implement the features yourself
- use a third-party library that has the desired features.

Often you can find a previous implementation of what you need. Since this is a common situation, there is a service supporting it: the [Python Package Index](#) (or PyPi for short).

Our task here is to install the autopep8 tool from PyPi. This will allow us to illustrate the use of virtual environments using the pyenv or virtualenv command, and installing and uninstalling PyPi packages using pip.

25.2 Using pip to Install Packages

Let's now look at another important tool for Python development: the Python Package Index, or PyPi for short. PyPi provides a large set of third-party python packages. If you want to do something in python, first check pypi, as often someone already ran into the problem and created a package solving it.

In order to install package from PyPi, use the pip command. We can search for PyPi for packages:

```
$ pip search --trusted-host pypi.python.org autopep8 pylint
```

It appears that the top two results are what we want so install them:

```
$ pip install --trusted-host pypi.python.org autopep8 pylint
```

This will cause pip to download the packages from PyPI, extract them, check their dependencies and install those as needed, then install the requested packages.

You can skip ‘--trusted-host pypi.python.org’ option if you have patched urllib3 on Python 2.7.9.

25.3 GUI

25.3.1 GUIZero

Install guizero with the following command:

```
sudo pip3 install guizero
```

For a comprehensive tutorial on guizero, [click here](#).

25.3.2 Kivy

You can install Kivy on OSX as follows:

```
brew install pkg-config sdl2 sdl2_image sdl2_ttf sdl2_mixer gstreamer
pip install -U Cython
pip install kivy
pip install pygame
```

A hello world program for kivy is included in the cloudmesh.robot repository. Which you can find here

- <https://github.com/cloudmesh/cloudmesh.robot/tree/master/projects/kivy>

To run the program, please download it or execute it in cloudmesh.robot as follows:

```
cd cloudmesh.robot/projects/kivy
python swim.py
```

To create stand alone packages with kivy, please see:

```
- https://kivy.org/docs/guide/packaging-osx.html
```

25.4 Formatting and Checking Python Code

First, get the bad code:

```
$ wget --no-check-certificate http://git.io/pXqb -O bad_code_example.py
```

Examine the code:

```
$ emacs bad_code_example.py
```

As you can see, this is very dense and hard to read. Cleaning it up by hand would be a time-consuming and error-prone process. Luckily, this is a common problem so there exist a couple packages to help in this situation.

25.5 Using autopep8

We can now run the bad code through autopep8 to fix formatting problems:

```
$ autopep8 bad_code_example.py >code_example_autopep8.py
```

Let us look at the result. This is considerably better than before. It is easy to tell what the `example1` and `example2` functions are doing.

It is a good idea to develop a habit of using `autopep8` in your python-development workflow. For instance: use `autopep8` to check a file, and if it passes, make any changes in place using the `-i` flag:

```
$ autopep8 file.py    # check output to see of passes
$ autopep8 -i file.py # update in place
```

If you use `pyCharm` you have the ability to use a similar function while pressing on `Inspect Code`.

25.6 Writing Python 3 Compatible Code

To write python 2 and 3 compatible code we recommend that you take a look at: http://python-future.org/compatible_idioms.html

25.7 Using Python on FutureSystems

This is only important if you use `FutureSystems` resources.

In order to use Python you must log into your `FutureSystems` account. Then at the shell prompt execute the following command:

```
$ module load python
```

This will make the `python` and `virtualenv` commands available to you.

The details of what the `module load` command does are described in the future lesson modules.

25.8 Ecosystem

25.8.1 pypi

Link: [pypi](#)

The Python Package Index is a large repository of software for the Python programming language containing a large number of packages [link]. The nice thing about `pypi` is that many packages can be installed with the program `'pip'`.

To do so you have to locate the `<package_name>` for example with the search function in `pypi` and say on the commandline:

```
pip install <package_name>
```

where `package_name` is the string name of the package. an example would be the package called `cloudmesh_client` which you can install with:

```
pip install cloudmesh_client
```

If all goes well the package will be installed.

25.8.2 Alternative Installations

The basic installation of python is provided by `python.org`. However others claim to have alternative environments that allow you to install python. This includes

- [Canopy](#)

- [Anaconda](#)
- [IronPython](#)

Typically they include not only the python compiler but also several useful packages. It is fine to use such environments for the class, but it should be noted that in both cases not every python library may be available for install in the given environment. For example if you need to use cloudmesh client, it may not be available as conda or Canopy package. This is also the case for many other cloud related and useful python libraries. Hence, we do recommend that if you are new to python to use the distribution form python.org, and use pip and virtualenv.

Additionally some python version have platform specific libraries or dependencies. For example coca libraries, .NET or other frameworks are examples. For the assignments and the projects such platform dependent libraries are not to be used.

If however you can write a platform independent code that works on Linux, OSX and Windows while using the python.org version but develop it with any of the other tools that is just fine. However it is up to you to guarantee that this independence is maintained and implemented. You do have to write requirements.txt files that will install the necessary python libraries in a platform independent fashion. The homework assignment PRG1 has even a requirement to do so.

In order to provide platform independence we have given in the class a ‘minimal’ python version that we have tested with hundreds of students: python.org. If you use any other version, that is your decision. Additionally some students not only use python.org but have used iPython which is fine too. However this class is not only about python, but also about how to have your code run on any platform. The homework is designed so that you can identify a setup that works for you.

However we have concerns if you for example wanted to use chameleon cloud which we require you to access with cloudmesh. cloudmesh is not available as conda, canopy, or other framework package. Cloudmesh client is available form pypi which is standard and should be supported by the frameworks. We have not tested cloudmesh on any other python version then python.org which is the open source community standard. None of the other versions are standard.

In fact we had students over the summer using canopy on their machines and they got confused as they now had multiple python versions and did not know how to switch between them and activate the correct version. Certainly if you know how to do that, than feel free to use canopy, and if you want to use canopy all this is up to you. However the homework and project requires you to make your program portable to python.org. If you know how to do that even if you use canopy, anaconda, or any other python version that is fine. Graders will test your programs on a python.org installation and not canpoy, anaconda, ironpython while using virtualenv. It is obvious why. If you do not know that answer you may want to think about that every time they test a program they need to do a new virtualenv and run vanilla python in it. If we were to run two instals in the same system, this will not work as we do not know if one student will cause a side effect for another. Thus we as instructors do not just have to look at your code but code of hundreds of students with different setups. This is a non scalable solution as every time we test out code from a student we would have to wipe out the OS, install it new, install an new version of whatever python you have elected, become familiar with that version and so on and on. This is the reason why the open source community is using python.org. We follow best practices. Using other versions is not a community best practice, but may work for an individual.

We have however in regards to using other python version additional bonus projects such as

- deploy run and document cloudmesh on ironpython
- deploy run and document cloudmesh on anaconda, develop script to generate a conda package form github

- deploy run and document cloudmesh on canopy, develop script to generate a conda package form github
- deploy run and document cloudmesh on ironpython
- other documentation that would be useful

25.9 Resources

If you are unfamiliar with programming in Python, we also refer you to some of the numerous online resources. You may wish to start with [Learn Python](#) or the book [Learn Python the Hard Way](#). Other options include [Tutorials Point](#) or [Code Academy](#), and the Python wiki page contains a long list of [references for learning](#) as well. Additional resources include:

- <https://virtualenvwrapper.readthedocs.io>
- <https://github.com/yyuu/pyenv>
- <https://amaral.northwestern.edu/resources/guides/pyenv-tutorial>
- <https://godjango.com/96-django-and-python-3-how-to-setup-pyenv-for-multiple-pythons/>
- <https://www.accelebrate.com/blog/the-many-faces-of-python-and-how-to-manage-them/>
- <http://ivory.idyll.org/articles/advanced-swc/>
- <http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html>
- <http://www.youtube.com/watch?v=0vJJlVBVTFg>
- <http://www.korokithakis.net/tutorials/python/>
- <http://www.afterhoursprogramming.com/tutorial/Python/Introduction/>
- <http://www.greenteapress.com/thinkpython/thinkCSpy.pdf>
- <https://docs.python.org/3.3/tutorial/modules.html>
- https://www.learnpython.org/en/Modules/_and/_Packages
- <https://docs.python.org/2/library/datetime.html>
- https://chrisalbon.com/python/strings/_to/_datetime.html

A very long list of useful information are also available from

- <https://github.com/vinta/awesome-python>
- https://github.com/rasbt/python_reference

This list may be useful as it also contains links to data visualization and manipulation libraries, and AI tools and libraries. Please note that for this class you can reuse such libraries if not otherwise stated.

25.9.1 Jupyter Notebook Tutorials

A Short Introduction to Jupyter Notebooks and NumPy To view the notebook, open this link in a background tab <https://nbviewer.jupyter.org/> and copy and paste the following link in the URL input area <https://cloudmesh.github.io/classes/lesson/prg/Jupyter-NumPy-tutorial-I523-F2017.ipynb> Then hit Go.

25.10 Exercises

Exercise 25.1 Write a python program called `iterate.py` that accepts an integer `n` from the command line. Pass this integer to a function called `iterate`.

The `iterate` function should then iterate from 1 to `n`. If the `i`-th number is a multiple of three, print “multiple of 3”, if a multiple of 5 print “multiple of 5”, if a multiple of both print “multiple of 3 and 5”, else print the value. ■

Exercise 25.2

1. Create a `pyenv` or `virtualenv` `~/ENV`
2. Modify your `~/.bashrc` shell file to activate your environment upon login.
3. Install the `docopt` python package using `pip`
4. Write a program that uses `docopt` to define a commandline program. Hint: modify the `iterate` program.
5. Demonstrate the program works and submit the code and output. ■

25.11 Python for Big Data

25.11.1 An Example with Pandas, NumPy and Matplotlib

In this example, we will download some traffic citation data for the city of Bloomington, IN, load it into Python and generate a histogram. In doing so, you will be exposed to important Python libraries for working with big data such as [numpy](#), [pandas](#) and [matplotlib](#).

Set Up Directories and Get Test Data

Data.gov is a government portal for open data and the [city of Bloomington, Indiana makes available a number of datasets there](#).

We will use traffic citations data for 2016.

To start, let’s create a separate directory for this project and download the CSV data:

```
1 $ cd ~/projects/i524
2 $ mkdir btown-citations
3 $ cd btown-citations
4 $ wget https://data.bloomington.in.gov/dataset/c543f0c1-1e37-46ce-a0ba-↵
   ↵ e0a949bd248a/resource/24841976-fd35-4483-a2b4-573bd1e77cfb/download↵
   ↵ /2016-first-quarter-citations.csv
```

Depending on your directory organization, the above might be slightly different for you.

If you go to the link to [data.gov](#) for Bloomington above, you will see that the citations data is organized per quarter, so there are a total of four files. Above, we downloaded the data for the first quarter. Go ahead and download the remaining three files with `wget`.

In this example, we will use three modules, `numpy`, `pandas` and `matplotlib`. If you set up `virtualenv` as described in the Python tutorial [<python_intro>](#), the first two of these are already installed for you. To install `matplotlib`, make sure you’ve activated your `virtualenv` and use `pip`:

```
5 $ source ~/ENV/bin/activate
6 $ pip install matplotlib
```


If you are using a different distribution of Python, you will need to make sure that all three of these modules are installed.

Load Data in Pandas

From the same directory where you saved the citations data, let's start the Python interpreter and load the citations data for Q1 2016

```
1 $ python
2 >>> from __future__ import division, print_function
3 >>> import numpy as np
4 >>> import pandas as pd
5 >>> import matplotlib.pyplot as plt
6 >>> data = pd.read_csv('2016-first-quarter-citations.csv')
```

If the first import statement seems confusing, take a look at the Python tutorial <python_intro>. The next three import statements load each of the modules we will use in this example. The final line uses Pandas' read_csv function to load the data into a Pandas DataFrame data structure.

Working with DataFrames

You can verify that you are working with a DataFrame and use some of its methods to take a look at the structure of the data as follows:

```
1 >>> type(data)
2 <class 'pandas.core.frame.DataFrame'>
3 >>> data.index
4 Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,
5 ...
6 197, 198, 199, 200, 201, 202, 203, 204, 205, 206],
7 dtype='int64', length=200)
8 >>> data.columns
9 Index([u'Citation Number', u'Date Issued', u'Time Issued', u'Location ',
10 u'District', u'Cited Person Age', u'Cited Person Sex',
11 u'Cited Person Race', u'Offense Code', u'Offense Description',
12 u'Officer Age', u'Officer Sex', u'Officer Race'],
13 dtype='object')
14 >>> data.dtypes
15 Citation Number          object
16 Date Issued              object
17 Time Issued              object
18 Location                 object
19 District                 object
20 Cited Person Age         float64
21 Cited Person Sex         object
22 Cited Person Race        object
23 Offense Code             object
24 Offense Description      object
25 Officer Age              float64
26 Officer Sex              object
27 Officer Race             object
28 dtype: object
29 >>> data.shape
30 (200, 15)
```

As you can see from the columns field, when the CSV file was read, the header line was used to

populate the name of the columns in the DataFrame. In addition, you will notice that `read_csv` correctly inferred the data type of some columns like *Age*, but not of others like *Date Issued* and *Time Issued*. `read_csv` is a very customizable function and in general, you can correct issues like this using the `dtype` and `converters` parameters. In this specific case, it makes more sense to combine the *Date Issued* and *Time Issued* columns into a new column containing a time stamp. We will see how to do this shortly.

You can also look at the data itself with the DataFrame's `head()` and `tail()` methods:

```
1 >>> data.head()
2 <Output omitted for brevity>
3 >>> data.tail()
4 <Output omitted for brevity>
```

In addition to letting you examine your data easily, DataFrames have methods that help you deal with missing values:

```
1 >>> data = data.dropna(how='any')
2 >>> data.shape
```

Adding columns to the data is also easy. Here, we add two columns. First, a `datetime` column that is a combination of the *Date Issued* and *Time Issued* columns originally in the data. Second, a column identifying what day of the week each citation was given. To understand this example better, take a look at the Python docs for the `strptime` and `strftime` functions in the `datetime` module linked above.

```
1 >>> from datetime import datetime
2 >>> data['DateTime Issued'] = data.apply(
3 ...     lambda row: datetime.strptime(row['Date Issued'] + ':' + row['Time ←
4 ...     ↪ Issued'], '%m/%d/%y:%I:%M %p'), axis=1
5 >>> data.columns
6 >>> data['Day of Week Issued'] = data.apply(
7 ...     lambda row: datetime.strftime(row['DateTime Issued'], '%A'), axis=1
8 ... )
```

Plotting with Matplotlib and NumPy

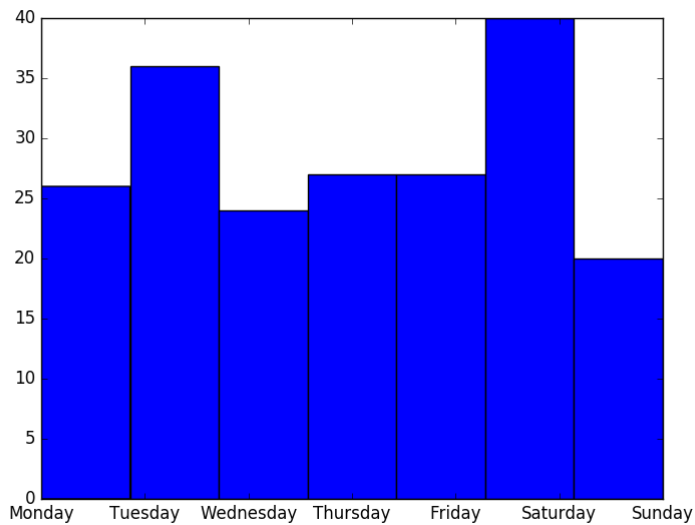
Let's say we want to see how many citations were given each day of the week. We gather the data first:

```
1 >>> days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', '←
2 ... ↪ Saturday', 'Sunday']
3 >>> dow_data = [days.index(dow) for dow in data['Day of Week Issued']]
4 >>> dow_data
5 <Output omitted for brevity>
```

Then we use `matplotlib` to plot it:

```
1 >>> fig = plt.figure()
2 >>> ax = fig.add_subplot(1, 1, 1)
3 >>> plt.hist(dow_data, bins=len(days))
4 >>> plt.xticks(range(len(days)), days)
5 >>> plt.show()
```

You should see something like this on your screen:



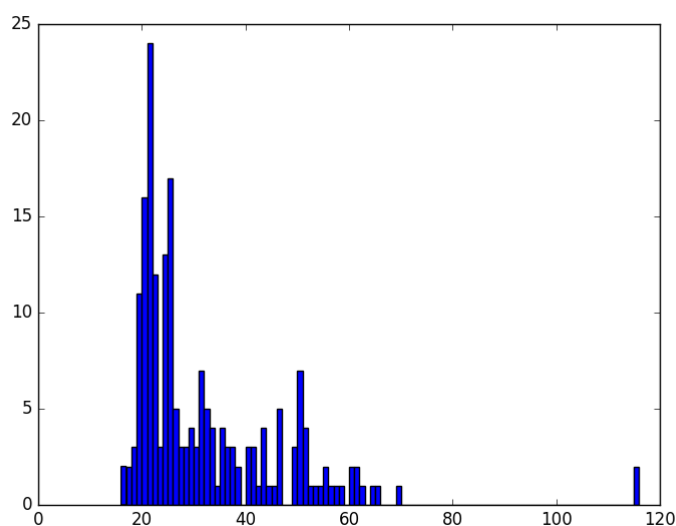
More DataFrame Manipulation and Plotting

DataFrames and numpy give us other ways to manipulate data. For example, we can plot a histogram of the ages of violators like this:

```

1 >>> ages = data['Cited Person Age'].astype(int)
2 >>> fig = plt.figure()
3 >>> ax = fig.add_subplot(1, 1, 1)
4 >>> plt.hist(ages, bins=np.max(ages) - np.min(ages))
5 >>> plt.show()

```



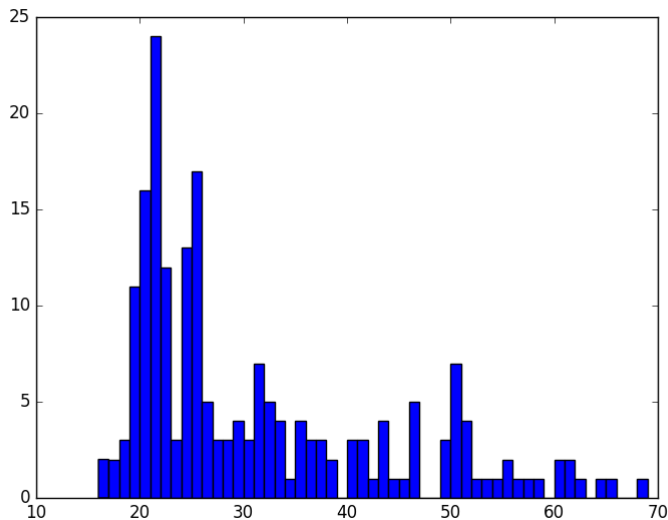
Surprisingly, we see some 116 year-old violators! This is probably an error in the data, so we can remove these data points easily and plot the histogram again:

```

1 >>> ages = ages[ages < 100]
2 >>> fig = plt.figure()
3 >>> ax = fig.add_subplot(1, 1, 1)

```

```
4 >>> plt.hist(ages, bins=np.max(ages) - np.min(ages))
5 >>> plt.show()
```



Saving Plots to PDF

Oftentimes, you will want to save your matplotlib graph as a PDF or an SVG file instead of just viewing it on your screen. For both, we need to create a figure and plot the histogram as before:

```
1 >>> fig = plt.figure()
2 >>> ax = fig.add_subplot(1, 1, 1)
3 >>> plt.hist(ages, bins=np.max(ages) - np.min(ages))
```

Then, instead of calling `plt.show()` we can invoke `plt.savefig()` to save as SVG:

```
1 >>> plt.savefig('hist.svg')
```

If we want to save the figure as PDF instead, we need to use the PdfPages module together with `savefig()`:

```
1 >>> import matplotlib.patches as mpatches
2 >>> from matplotlib.backends.backend_pdf import PdfPages
3 >>> pp = PdfPages('hist.pdf')
4 >>> fig.savefig(pp, format='pdf')
5 >>> pp.close()
```

Next Steps and Exercises

There is a lot more to working with pandas, numpy and matplotlib than we can show you here, but hopefully this example has piqued your curiosity.

Don't worry if you don't understand everything in this example. For a more detailed explanation on these modules and the examples we did, please take a look at the tutorials below. The numpy and pandas tutorials are mandatory if you want to be able to use these modules, and the matplotlib gallery has many useful code examples.

25.11.2 Summary of Useful Libraries

Numpy

- <http://www.numpy.org/>

According to the Numpy Web page “NumPy is a package for scientific computing with Python. It contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, useful linear algebra, Fourier transform, and random number capabilities”.

Tutorial: <https://docs.scipy.org/doc/numpy-dev/user/quickstart.html>

Matplotlib

- <http://matplotlib.org/>

According to the Matplotlib Web page, “matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell (ala MATLAB* or Mathematica), web application servers, and six graphical user interface toolkits.”

Matplotlib Gallery: <http://matplotlib.org/gallery.html>

Pandas

- <http://pandas.pydata.org/>

According to the Pandas Web page, “Pandas is a library library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.”

In addition to access to charts via matplotlib it has elementary functionality for conduction data analysis. Pandas may be very suitable for your projects.

Tutorial: <http://pandas.pydata.org/pandas-docs/stable/10min.html>

Pandas Cheat Sheet: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

25.11.3 Big Data Libraries

Scipy

- <https://www.scipy.org/>

According to the Web page, SciPy (pronounced *Sigh Pie*) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

- NumPy
- IPython
- Pandas
- Matplotlib
- Sympy
- SciPy library

It is thus an agglomeration of useful packages and will probably suffice for your projects in case you use Python.

ggplot

- <http://ggplot.yhathq.com/>

According to the ggplot python Web page ggplot is a plotting system for Python based on R's ggplot2. It allows to quickly generate some plots quickly with little effort. Often it may be easier to use than matplotlib directly.

seaborn

- http://www.data-analysis-in-python.org/t_seaborn.html

The good library for plotting is called seaborn which is build on top of matplotlib. It provides high level templates for common statistical plots.

- Gallery: <http://stanford.edu/~mwaskom/software/seaborn/examples/index.html>
- Original Tutorial: <http://stanford.edu/~mwaskom/software/seaborn/tutorial.html>
- Additional Tutorial: <https://stanford.edu/~mwaskom/software/seaborn/tutorial/distributions.html>

Here are some examples from a previous class:

- <https://github.com/bigdata-i523/hid231/blob/master/experiment/seaborn/seaborn-exercises.ipynb>
- https://github.com/bigdata-i523/hid231/blob/master/experiment/learning-jupyter/learning_jupyter_notebook.ipynb

Exercise 25.3 Take these examples and create sections in latex that can be added to the book. Describe the process.

1. export the ipynb as rst 2. use pandoc to export it to tex 3. do some cleanup on the tex files

Can this be automated with a cmd5 script such as

```
7 cms ipynb [url=URL | file=FILE] --output FILENAME
```

Bokeh

Bokeh is an interactive visualization library with focus on web browsers for display. Its goal is to provide a similar experience as D3.js

- URL: <http://bokeh.pydata.org/>
- Gallery: <http://bokeh.pydata.org/en/latest/docs/gallery.html>

pygal

Pygal is a simple API to produce graphs that can be easily embedded into your Web pages. It contains annotations when you hover over data points. It also allows to present the data in a table.

- <http://pygal.org/>

Network and Graphs

- igraph: http://www.pythonforsocialscientists.org/t_igraph.html
- networkx: <https://networkx.github.io/>

REST

- django REST FFramework <http://www.django-rest-framework.org/>
- flask <https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask>
- requests <https://realpython.com/blog/python/api-integration-in-python/>
- urllib2 <http://rest.elkstein.org/2008/02/using-rest-in-python.html> (not recommended)
- web <http://www.dreamsyssoft.com/python-scripting-tutorial/create-simple-rest-web-service-with-python.php> (not recommended)
- bottle <http://bottlepy.org/docs/dev/index.html>
- falcon <https://falconframework.org/>
- eve <http://python-eve.org/>
- <https://code.tutsplus.com/tutorials/building-rest-apis-using-eve--cms-22961>

25.12 Parsing Data

Being able to parse data is an important activity in the data analysis process. Not all data may be following a specific format and the data may need to be extracted.

25.12.1 notebook.md Parser

We are using a notebook.md to communicate what students have done throughout the semester. We like to make a simple cmd5 command that parses the notebook.md file and check it upon correctness.

An example for a notebook.md file is located here

- <https://raw.githubusercontent.com/bigdata-i523/sample-hid000/master/notebook.md>

The following code may inspire you

- <https://github.com/bigdata-i523/hid203/tree/master/experiment>

We like to implement the following functionality and use docopts to document the command.

```

1 cms class notebook [--git=GITREPONAME] --verify hid
2
3     verifies the correctness of the notebook.md file
4
5 cms class notebook [--git=GITREPONAME] --log
6
7     displays the log of the notebook.md
8
9 cms class notebook [--git=GITREPONAME] --history
10
11     displays a true or false for each week since the first occurrence
12     of the notebook.md file in the git repository

```

Exercise 25.4 Write a notebook.md parser

Exercise 25.5 How can this command be generalized to provide not only information for a student but to provide information for the class. Example: can we identify preferred days of when the notebooks are checked in. Can we identify a list of students that have not updated the notebook for a week? Can we identify the list of student s that have updated the notebook for a week? ■

25.12.2 Video Length

The Latex source of this class contains a macro to include videos.

Given a \LaTeX file, can you create a table that includes the names of all videos in that file and sums up the total viewing time. Previously the document was stored in RST and the code from a previous student may inspire you. Can you recreate it for \LaTeX ?

- <https://github.com/bigdata-i523/hid107/blob/master/cloudmesh/bar/command/mycommand.py>

```
1 cms class video list FILENAME --output=[tabular|longtable|csv|txt]
2
3 prints the videolist in the given format. txt means it is just ASCII
```

Exercise 25.6 Write a tool that extracts the information for video length. ■

Exercise 25.7 Write a tool that finds all youtube urls that are not in a video latex macro. ■

25.12.3 Dask

Many times operations need to be done on data in parallel to utilize modern processor architectures.

Dask provides a *dynamic task scheduling* which is optimized for computation. It is similar to other frameworks such as Airflow, Luigi, Celery, or Make. However it is specializing in optimized interactive computational workloads.

Furthermore, Dask targets Big Data *collections* such as parallel arrays, dataframes, and lists. These collections are commonly found in NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments. While using the Dask implementation we can replace the original imports from the appropriate framework, replace them with Dask imports and implicitly use parallel collections that utilize internally the dynamic task schedulers.

More information can be found at:

- <https://dask.pydata.org>

Exercise 25.8 Conduct a performance study that showcases the difference of doing parallel calculations in Dask, calculations in a framework such as SciPy, and regular unthreaded python code. ■

Python - Advanced

26	Numpy	339
26.1	Float Range	
26.2	Arrays	
26.3	Array Operations	
26.4	Linear Algebra	
26.5	Resources	
27	Scipy	343
27.1	Introduction	
27.2	References	
28	OpenCV	349
28.1	Overview	
28.2	Installation	
28.3	A Simple Example	
28.4	Additional Features	
28.5	Secchi Disk	
28.6	Setup for OSX	
28.7	Black and white	
29	NIST Pedestrian and Face Detection .	359
30	Python Fingerprint Example	371
30.1	Overview	
30.2	Objectives	
30.3	Prerequisites	
30.4	Implementation	
30.5	Utility functions	
30.6	Dataset	
30.7	Data Model	
30.8	Plotting	
30.9	Putting it all Together	



26. Numpy

NumPy is a popular library on that is used by many other python librariessuch as pandas, and SciPy. It provides simple to use array operations for data. This helps to access arrays in a more intuitive fashion and introduces various matrix operations.

We provide a short introduction to Numpy.

First we import the modules needed for this introduction and abreviate them with the as feature of the import statement

```
1 import numpy as np
2 import matplotlib as mpl
3 import matplotlib.pyplot as plt
```

Now we showcase some features of Numpy.

26.1 Float Range

`arange()` is like `range()`, but for floating-point numbers.

```
1 X = np.arange(0.2, 1, .1)
```

```
1 print (X)
```

We use this function to generate parameter space that can then be iterated on.

```
1 P = 10.0 ** np.arange(-7, 1, 1)
2
3 print (P)
```

```
1 for x, p in zip(X, P):
2     print ('%f, %f' % (x, p))
```

26.2 Arrays

To create one dimensional arrays you use

```
1 a = np.array([1, 2, 3])
```

To check some properties you can use the following

```
1 print(type(a))          # Prints "<class 'numpy.ndarray'>"
```

```
1 print(a.shape)         # Prints "(3,)"
```

The shape indicates that in the first dimension, there are 3 elements. To print the actual values you can use

```
1 print(a)               # Prints the values of the array
```

```
2 print(a[0], a[1], a[2]) # Prints "1 2 3"
```

To change values you can use the index of the element or use any other python method to do so. IN our example we change the first element to 42

```
1 a[0] = 42
```

```
2 print(a)
```

To create more dimensional arrays you use

```
1 b = np.array([[1,2,3],[4,5,6]]) # Create a 2 dimensional array
```

```
2 print(b.shape)                 # Prints "(2, 3)"
```

```
3 print(b[0, 0], b[0, 1], b[1, 0]) # Prints "1 2 4"
```

26.3 Array Operations

Let us first create some arrays with a predefined datatype

```
1 x = np.array([[1,2],[3,4]], dtype=np.float64)
```

```
2 y = np.array([[5,6],[7,8]], dtype=np.float64)
```

```
1 print(x)
```

```
1 print(y)
```

To add the numbers use

```
1 print(x+y)
```

Other functions such as -, *, / behave as expected using elementwise operations:

```
1 print(x-y)
```

```
1 print(x*y)
```

```
1 print(x/y)
```

To apply functions such as 'sin' make sure you use the function provided by the numpy package such as `np.sin`. The list of functions is included in the manual at * <https://docs.scipy.org/doc/numpy/reference/routines.math.html>

```
1 print(np.sin(x))
```

```
1 print (np.sum(x))
```

Computations can also be applied to columns and rows

```
1 print(np.sum(x, axis=0)) # sum of each column
2 print(np.sum(x, axis=1)) # sum of each row
```

26.4 Linear Algebra

Linear algebra methods are also provided.

```
1 from numpy import linalg
2
3 A = np.diag((1,2,3))
4
5 w,v = linalg.eig(A)
6
7 print ('w =', w)
8 print ('v =', v)
```

26.5 Resources

- <https://docs.scipy.org/doc/numpy/>
- <http://cs231n.github.io/python-numpy-tutorial/#numpy>



27. Scipy

SciPy is a library built above numpy and has a number of off the shelf algorithms and operations implemented. These include algorithms from calculus (such as integration), statistics, linear algebra, image-processing, signal processing, machine learning.

To achieve this, SciPy bundles a number of useful open-source software for mathematics, science, and engineering. It includes the following packages:

NumPy, for managing N-dimensional arrays

SciPy library, to access fundamental scientific computing capabilities

Matplotlib, to conduct 2D plotting

IPython, for an Interactive console (see jupyter)

Sympy, for symbolic mathematics

pandas, for providing data structures and analysis

27.1 Introduction

First we add the usual scientific computing modules with the typical abbreviations, including sp for scipy. We could invoke scipy's statistical package as sp.stats, but for the sake of laziness we abbreviate that too.

```
import numpy as np # import numpy
import scipy as sp # import scipy
from scipy import stats # refer directly to stats rather than sp.stats
import matplotlib as mpl # for visualization
from matplotlib import pyplot as plt # refer directly to pyplot
# rather than mpl.pyplot
```

Now we create some random data to play with. We generate 100 samples from a Gaussian distribution centered at zero.

```
s = sp.randn(100)
```

How many elements are in the set?

```
print ('There are',len(s),'elements in the set')
```

What is the mean (average) of the set?

```
print ('The mean of the set is',s.mean())
```

What is the minimum of the set?

```
print ('The minimum of the set is',s.min())
```

What is the maximum of the set?

```
print ('The maximum of the set is',s.max())
```

We can use the scipy functions too. What's the median?

```
print ('The median of the set is',sp.median(s))
```

What about the standard deviation and variance?

```
print ('The standard deviation is',sp.std(s),
      'and the variance is',sp.var(s))
```

Isn't the variance the square of the standard deviation?

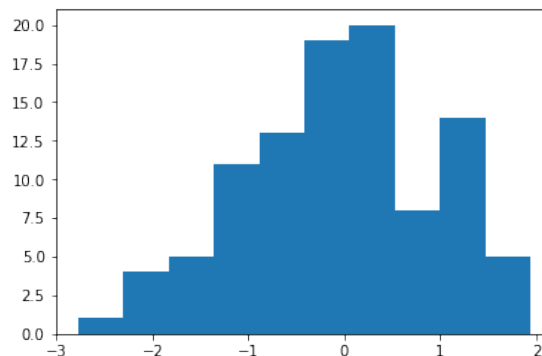
```
print ('The square of the standard deviation is',sp.std(s)**2)
```

How close are the measures? The differences are close as the following calculation shows

```
print ('The difference is',abs(sp.std(s)**2 - sp.var(s)))
print ('And in decimal form, the difference is %0.16f' %
      (abs(sp.std(s)**2 - sp.var(s))))
```

How does this look as a histogram?

```
plt.hist(s) # yes, one line of code for a histogram
plt.show()
```

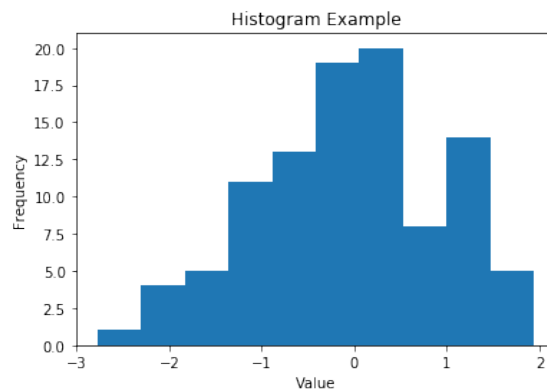


Let's add some titles.

```
plt.clf() # clear out the previous plot

plt.hist(s)
plt.title("Histogram Example")
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.show()
```

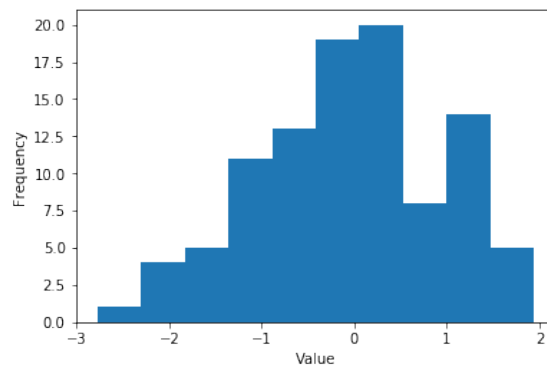


Typically we do not include titles when we prepare images for inclusion in LaTeX. There we use the caption to describe what the figure is about.

```
plt.clf() # clear out the previous plot

plt.hist(s)
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.show()
```



Let's try out some linear regression, or curve fitting.

```
import random

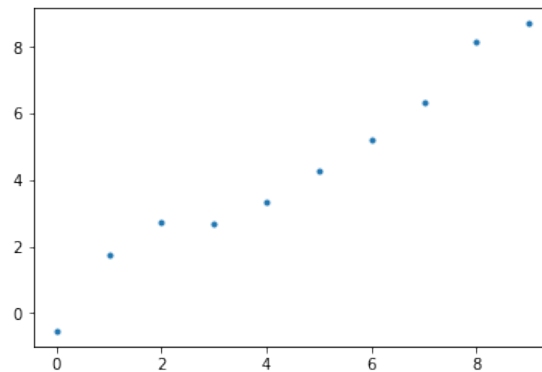
def F(x):
    return 2*x - 2

def add_noise(x):
    return x + random.uniform(-1,1)

X = range(0,10,1)

Y = []
for i in range(len(X)):
    Y.append(add_noise(X[i]))

plt.clf() # clear out the old figure
plt.plot(X,Y,'.')
plt.show()
```



Now let's try linear regression to fit the curve.

```
m, b, r, p, est_std_err = stats.linregress(X,Y)
```

What is the slope and y-intercept of the fitted curve?

```
print ('The slope is',m,'and the y-intercept is', b)

def Fprime(x): # the fitted curve
    return m*x + b
```

Now let's see how well the curve fits the data. We'll call the fitted curve F'.

```
X = range(0,10,1)

Yprime = []
for i in range(len(X)):
    Yprime.append(Fprime(X[i]))

plt.clf() # clear out the old figure

# the observed points, blue dots
plt.plot(X, Y, '.', label='observed points')

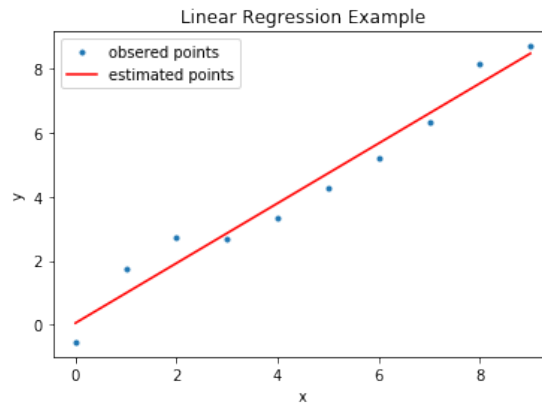
# the interpolated curve, connected red line
plt.plot(X, Yprime, 'r-', label='estimated points')

plt.title("Linear Regression Example") # title
plt.xlabel("x") # horizontal axis title
plt.ylabel("y") # vertical axis title
# legend labels to plot
plt.legend(['observed points', 'estimated points'])

# comment out so that you can save the figure
#plt.show()
```

To save images into a PDF file for inclusion into $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ documents you can save the images as follows. Other formats such as png are also possible, but the quality is naturally not sufficient for inclusion in papers and documents. For that you certainly want to use PDF. The save of the figure has to occur before you use the `show()` command.

```
plt.savefig("regression.pdf", bbox_inches='tight')
plt.savefig('regression.png')
plt.show()
```

27.2 References

For more information about SciPy we recommend that you visit the following link

<https://www.scipy.org/getting-started.html#learning-to-work-with-scipy>

Additional material and inspiration for this section are from

- “Getting Started guide” <https://www.scipy.org/getting-started.html>
- Prasanth. “Simple statistics with SciPy.” Comfort at 1 AU. February 28, 2011. <https://oneau.wordpress.com/2011/02/28/simple-statistics-with-scipy/>.
- SciPy Cookbook. Lasted updated: 2015. <http://scipy-cookbook.readthedocs.io/>.



28. OpenCV

OpenCV (Open Source Computer Vision Library) is a library of thousands of algorithms for various applications in computer vision and machine learning. It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. In this tutorial, we will explain basic features of this library, including the implementation of a simple example.

28.1 Overview

OpenCV has countless functions for image and videos processing. The pipeline starts with reading the images, low-level operations on pixel values, preprocessing e.g. denoising, and then multiple steps of higher-level operations which vary depending on the application. OpenCV covers the whole pipeline, especially providing a large set of library functions for high-level operations. A simpler library for image processing in Python is Scipy's multi-dimensional image processing package (`scipy.ndimage`).

28.2 Installation

OpenCV for Python can be installed on Linux in multiple ways, namely PyPI(Python Package Index), Linux package manager (`apt-get` for Ubuntu), Conda package manager, and also building from source. You are recommended to use PyPI. Here's the command that you need to run:

```
pip install opencv-python
```

This was tested on Ubuntu 16.04 with a fresh Python 3.6 virtual environment. In order to test, import the module in Python command line:

```
>>> import cv2
```

If it does not raise an error, it is installed correctly. Otherwise, try to solve the error.

For installation on Windows, see:

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html#install-opencv-python-in-windows

Note that building from source can take a long time and may not be feasible for deploying to limited platforms such as Raspberry Pi.

28.3 A Simple Example

In this example, an image is loaded. A simple processing is performed, and the result is written to a new image.

28.3.1 Loading an image

```
%matplotlib inline
import cv2

img = cv2.imread('opencv_files/4.2.01.tiff')
```

The image was downloaded from USC standard database:

- <http://sipi.usc.edu/database/database.php?volume=misc&image=9>

28.3.2 Displaying the image

The image is saved in a numpy array. Each pixel is represented with 3 values (R,G,B). This provides you with access to manipulate the image at the level of single pixels. You can display the image using `imshow` function as well as Matplotlib's `imshow` function.

You can display the image using `imshow` function:

```
cv2.imshow('Original',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

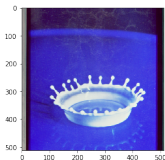
or you can use Matplotlib. If you have not installed Matplotlib before, install it using:

```
pip install matplotlib
```

Now you can use:

```
import matplotlib.pyplot as plt
plt.imshow(img)
```

which results in



28.3.3 Scaling and Rotation

Scaling (resizing) the image relative to different axis

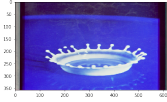
```
res = cv2.resize(img,
                  None,
                  fx=1.2,
```

```

fy=0.7,
interpolation=cv2.INTER_CUBIC)
plt.imshow(res)

```

which results in



Rotation of the image for an angle of t

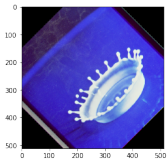
```

rows,cols,_ = img.shape
t = 45
M = cv2.getRotationMatrix2D((cols/2,rows/2),t,1)
dst = cv2.warpAffine(img,M,(cols,rows))

plt.imshow(dst)

```

which results in



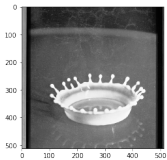
28.3.4 Gray-scaling

```

img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img2, cmap='gray')

```

which results in



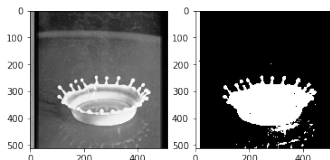
28.3.5 Image Thresholding

```

ret,thresh = cv2.threshold(img2,127,255,cv2.THRESH_BINARY)
plt.subplot(1,2,1), plt.imshow(img2, cmap='gray')
plt.subplot(1,2,2), plt.imshow(thresh, cmap='gray')

```

which results in



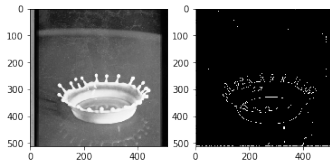
28.3.6 Edge Detection

Edge detection using Canny edge detection algorithm

```
edges = cv2.Canny(img2,100,200)

plt.subplot(121),plt.imshow(img2,cmap = 'gray')
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
```

which results in



28.4 Additional Features

OpenCV has implementations of many machine learning techniques such as KMeans and Support Vector Machines, that can be put into use with only a few lines of code. It also has functions especially for video analysis, feature detection, object recognition and many more. You can find out more about them in their website

- <https://docs.opencv.org/3.0-beta/index.html>

OpenCV was initially developed for C++ and still has a focus on that language, but it is still one of the most valuable image processing libraries in Python.

28.5 Secchi Disk

28.5.1 Overview

More information about Secchi Disk can be found at:

- https://en.wikipedia.org/wiki/Secchi/_disk

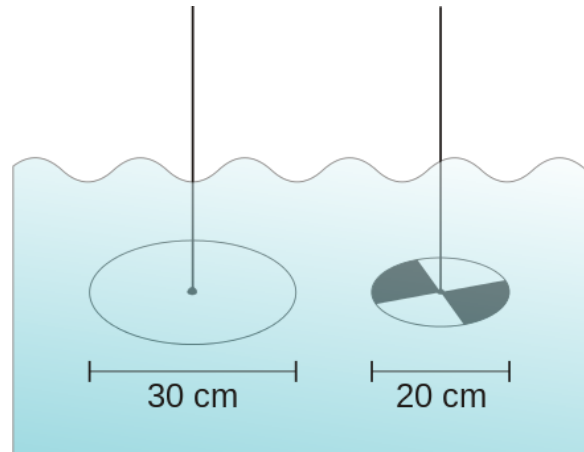


Figure 28.1: secchi

Figure: Different kinds of Secchi disks. A marine style on the left and the freshwater version on the right [wikipedia]

28.6 Setup for OSX

First let's setup the environment for OSX

```
import os, sys
from os.path import expanduser
os.path
home = expanduser("~")
sys.path.append('/usr/local/Cellar/opencv/3.3.1_1/lib/python3.6/site-packages/')
sys.path.append(home + '/.pyenv/versions/OPENCV/lib/python3.6/site-packages/')
import cv2
cv2.__version__
! pip install numpy > tmp.log
! pip install matplotlib >> tmp.log
%matplotlib inline
```

28.6.1 Step 1: Record the video

Record the video on the robot

28.6.2 Step 2: Analyse the images from the Video

For now we just selected 4 images from the video

```
import cv2
import matplotlib.pyplot as plt

img1 = cv2.imread('secchi/secchi1.png')
```

```
img2 = cv2.imread('secchi/secchi2.png')
img3 = cv2.imread('secchi/secchi3.png')
img4 = cv2.imread('secchi/secchi4.png')

figures = []
fig = plt.figure(figsize=(18, 16))
for i in range(1,13):
    figures.append(fig.add_subplot(4,3,i))
count = 0
for img in [img1,img2,img3,img4]:
    figures[count].imshow(img)

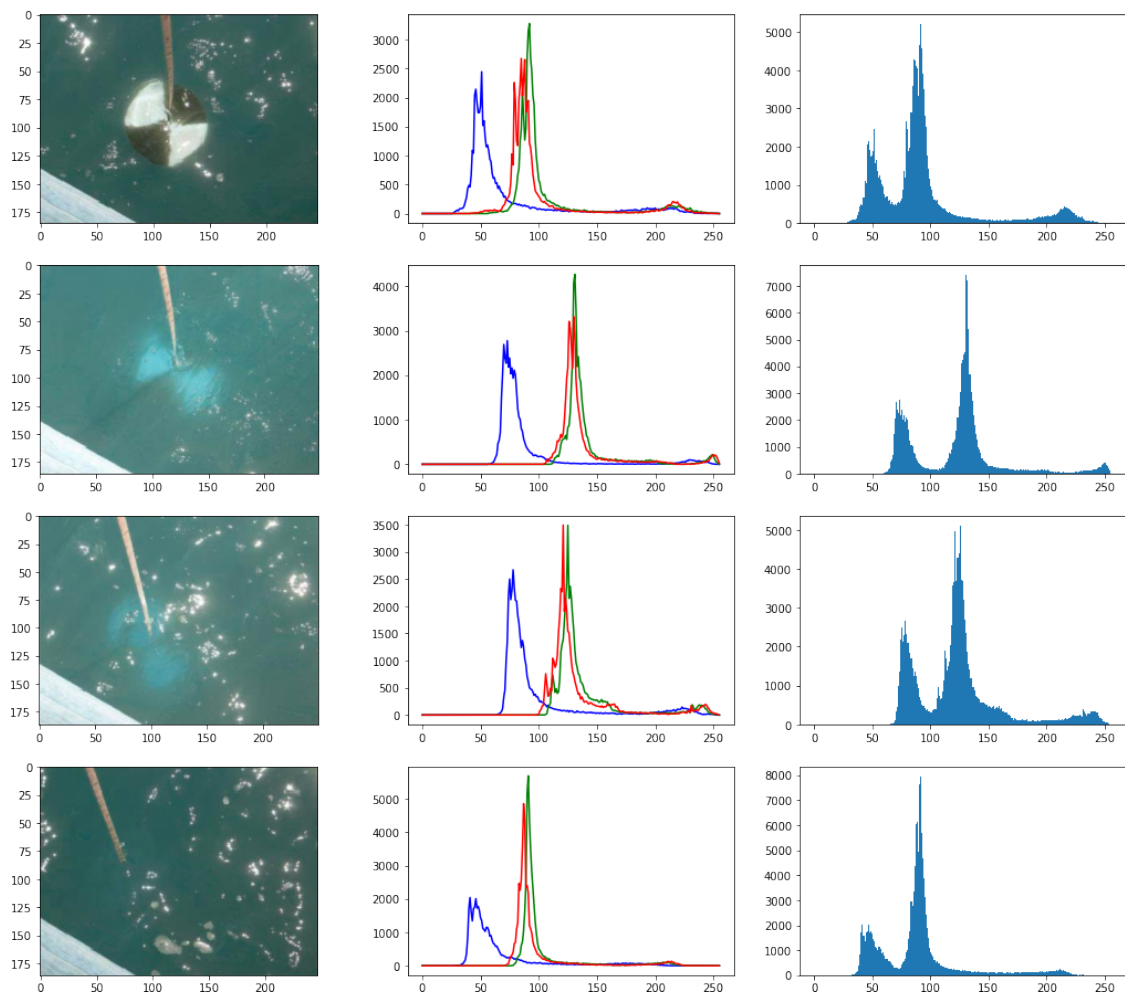
    color = ('b','g','r')
    for i,col in enumerate(color):
        histr = cv2.calcHist([img],[i],None,[256],[0,256])
        figures[count+1].plot(histr,color = col)

    figures[count+2].hist(img.ravel(),256,[0,256])

    count += 3

print("Legend")
print("First column = image of Secchi disk")
print("Second column = histogram of colors in image")
print("Third column = histogram of all values")

plt.show()
```

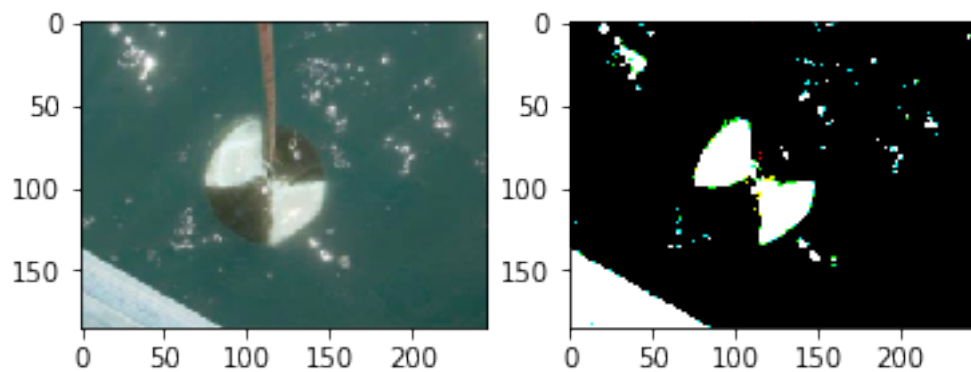



Rotation of the image for an angle of t

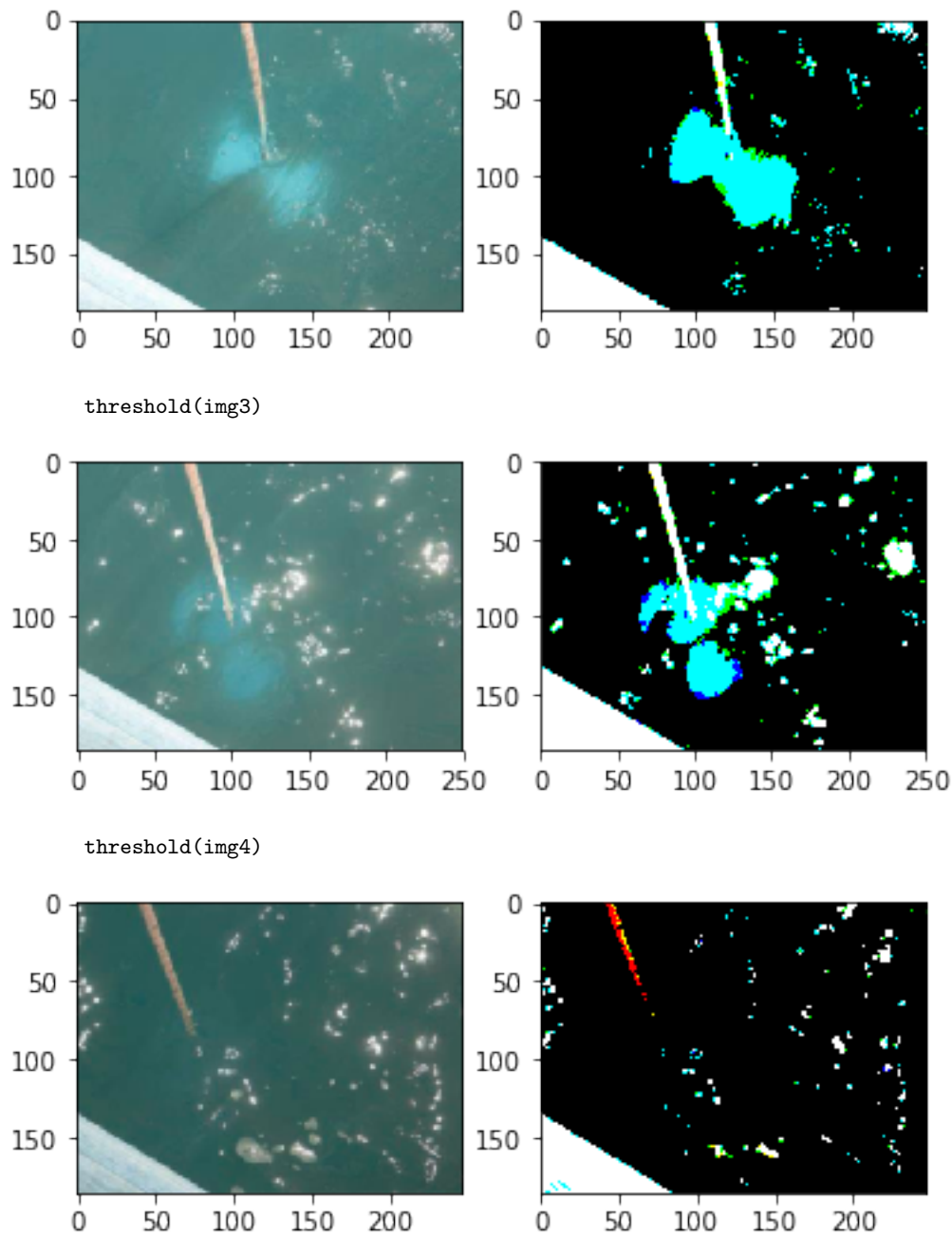
Image Thresholding

```
def threshold(img):
    ret,thresh = cv2.threshold(img,150,255,cv2.THRESH_BINARY)
    plt.subplot(1,2,1), plt.imshow(img, cmap='gray')
    plt.subplot(1,2,2), plt.imshow(thresh, cmap='gray')

threshold(img1)
```



```
threshold(img2)
```

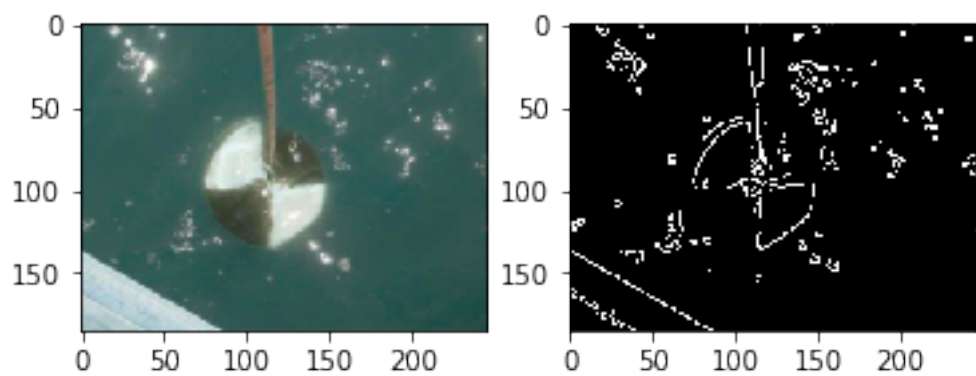


Edge Detection

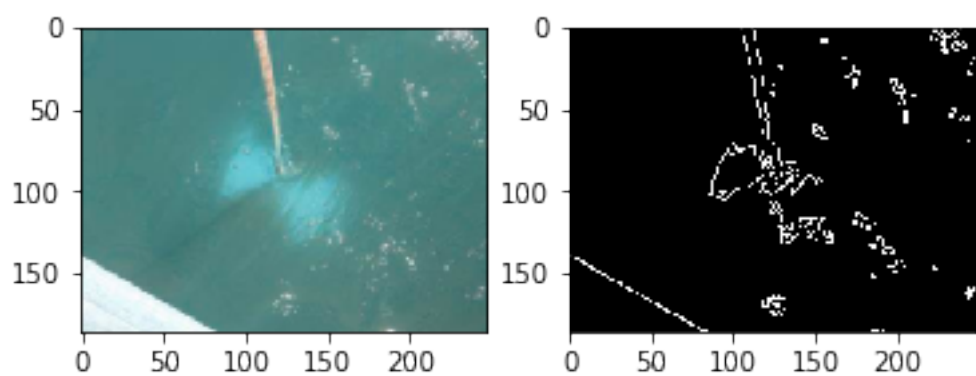
Edge detection using Canny edge detection algorithm

```
def find_edge(img):
    edges = cv2.Canny(img,50,200)
    plt.subplot(121),plt.imshow(img,cmap = 'gray')
    plt.subplot(122),plt.imshow(edges,cmap = 'gray')

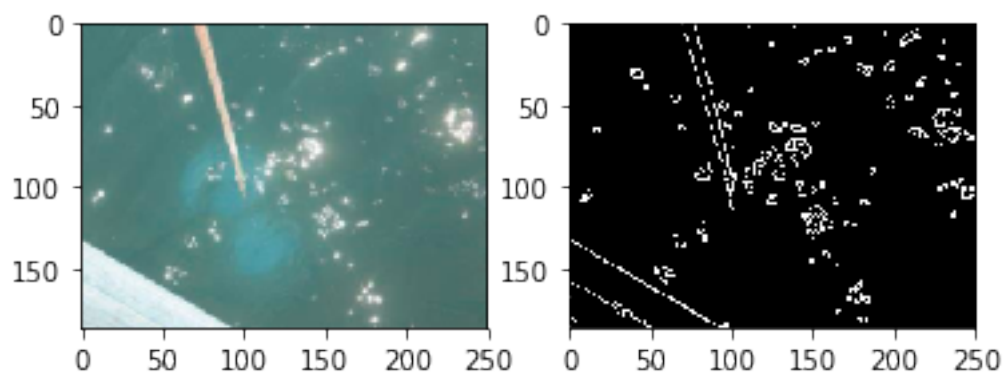
find_edge(img1)
```



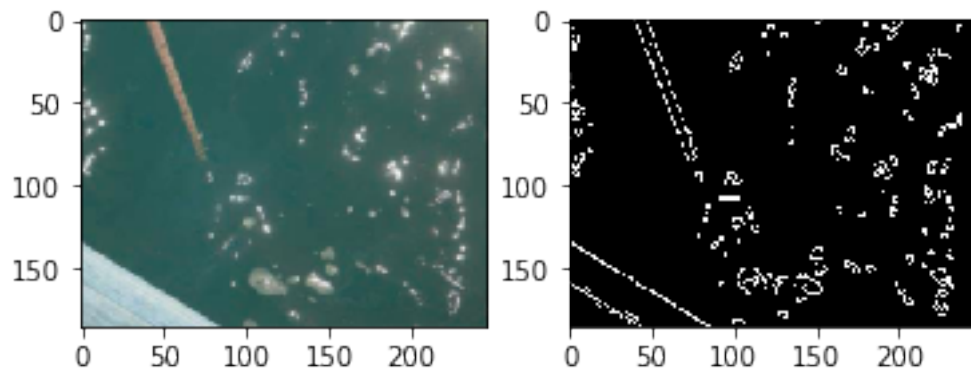
```
find_edge(img2)
```



```
find_edge(img3)
```

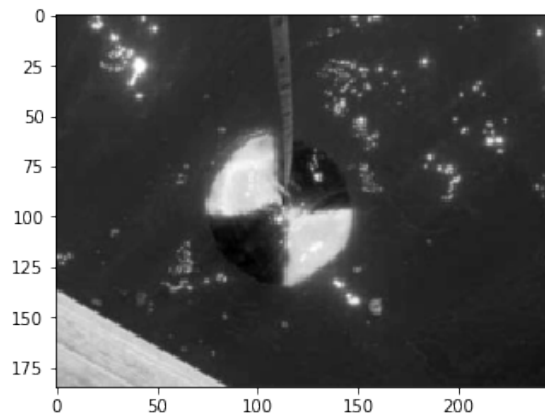


```
find_edge(img4)
```



28.7 Black and white

```
bw1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
plt.imshow(bw1, cmap='gray')
```



29. NIST Pedestrian and Face Detection

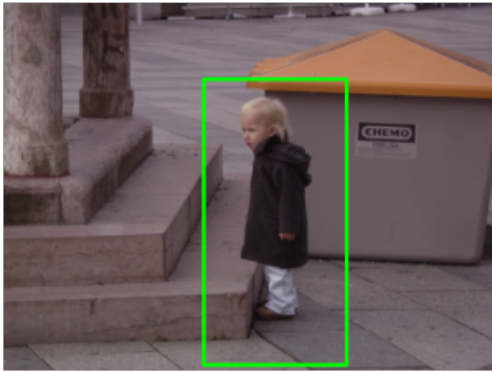
Pedestrian and Face Detection uses OpenCV to identify people standing in a picture or a video and NIST use case in this document is built with Apache Spark and Mesos clusters on multiple compute nodes.

The example in this tutorial deploys software packages on OpenStack using Ansible with its roles.

Original:



Pedestrian Detected:



Original



Pedestrian and Face/eyes Detected



29.0.1 Introduction

Human (pedestrian) detection and face detection have been studied during the last several years and models for them have improved along with Histograms of Oriented Gradients (HOG) for Human Detection [1]. OpenCV is a Computer Vision library including the SVM classifier and the HOG object detector for pedestrian detection and INRIA Person Dataset [2] is one of popular samples for both training and testing purposes. In this document, we deploy Apache Spark on Mesos clusters to train and apply detection models from OpenCV using Python API.

INRIA Person Dataset

This dataset contains positive and negative images for training and test purposes with annotation files for upright persons in each image. 288 positive test images, 453 negative test images, 614 positive training images and 1218 negative training images are included along with normalized 64x128 pixel formats. 970MB dataset is available to download [3].

HOG with SVM model

Histogram of Oriented Gradient (HOG) and Support Vector Machine (SVM) are used as object detectors and classifiers and built-in python libraries from OpenCV provide these models for human detection.

Ansible Automation Tool

Ansible is a python tool to install/configure/manage software on multiple machines with JSON files where system descriptions are defined. There are reasons why we use Ansible:

- Expandable: Leverages Python (default) but modules can be written in any language
- Agentless: no setup required on managed node
- Security: Allows deployment from user space; uses ssh for authentication
- Flexibility: only requires ssh access to privileged user
- Transparency: YAML Based script files express the steps of installing and configuring software
- Modularity: Single Ansible Role (should) contain all required commands and variables to deploy software package independently
- Sharing and portability: roles are available from source (github, bitbucket, gitlab, etc) or the Ansible Galaxy portal

We use Ansible roles to install software packages for Humand and Face Detection which requires to run OpenCV Python libraries on Apache Mesos with a cluster configuration. Dataset is also downloaded from the web using an ansible role.

29.0.2 Deployment by Ansible

Ansible is to deploy applications and build clusters for batch-processing large datasets towards target machines e.g. VM instances on OpenStack and we use ansible roles with *include* directive to organize layers of big data software stacks (BDSS). Ansible provides abstractions by Playbook Roles and reusability by Include statements. We define X application in X Ansible Role, for example, and use include statements to combine with other applications e.g. Y or Z. The layers exist in sub directories (see below) to add modularity to your Ansible deployment. For example, there are five roles used in this example that are Apache Mesos in a scheduler layer, Apache Spark in a processing layer, a OpenCV library in an application layer, INRIA Person Dataset in a dataset layer and a python script for human and face detection in an analytics layer. If you have an additional software package to add, you can simply add a new role in a main ansible playbook with *include* directive. With this, your Ansible playbook maintains simple but flexible to add more roles without having a large single file which is getting difficult to read when it deploys more applications on multiple layers. The main Ansible playbook runs Ansible roles in order which look like:

```
1 ---
2 include: sched/00-mesos.yml
3 include: proc/01-spark.yml
```

```

4 include: apps/02-opencv.yml
5 include: data/03-inria-dataset.yml
6 Include: anlys/04-human-face-detection.yml

```

Directory names e.g. sched, proc, data, or anlys indicate BDSS layers like: - sched: scheduler layer - proc: data processing layer - apps: application layer - data: dataset layer - anlys: analytics layer and two digits in the filename indicate an order of roles to be run.

29.0.3 Cloudmesh for Provisioning

It is assumed that virtual machines are created by cloudmesh, the cloud management software. For example on OpenStack,

```
cm cluster create -N=6
```

command starts a set of virtual machine instances. The number of machines and groups for clusters e.g. namenodes and datanodes are defined in the Ansible inventory file, a list of target machines with groups, which will be generated once machines are ready to use by cloudmesh. Ansible roles install software and dataset on virtual clusters after that stage.

29.0.4 Roles Explained for Installation

Mesos role is installed first as a scheduler layer for masters and slaves where mesos-master runs on the masters group and mesos-slave runs on the slaves group. Apache Zookeeper is included in the mesos role therefore mesos slaves find an elected mesos leader for the coordination. Spark, as a data processing layer, provides two options for distributed job processing, batch job processing via a cluster mode and real-time processing via a client mode. The Mesos dispatcher runs on a masters group to accept a batch job submission and Spark interactive shell, which is the client mode, provides real-time processing on any node in the cluster. Either way, Spark is installed later to detect a master (leader) host for a job submission. Other roles for OpenCV, INRIA Person Dataset and Human and Face Detection Python applications are followed by.

The following software are expected in the stacks according to the [github](#):

- mesos cluster (master, worker)
- spark (with dispatcher for mesos cluster mode)
- openCV
- zookeeper
- INRIA Person Dataset
- Detection Analytics in Python
- [1] Dalal, Navneet, and Bill Triggs. “Histograms of oriented gradients for human detection.” 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05). Vol. 1. IEEE, 2005. [pdf]
- [2] <http://pascal.inrialpes.fr/data/human/>
- [3] <ftp://ftp.inrialpes.fr/pub/lear/douze/data/INRIAPerson.tar>
- [4] <https://docs.python.org/2/library/configparser.html>

Server groups for Masters/Slaves by Ansible inventory

We may separate compute nodes in two groups: masters and workers therefore Mesos masters and zookeeper quorums manage job requests and leaders and workers run actual tasks. Ansible needs

group definitions in their inventory therefore software installation associated with a proper part can be completed.

Example of Ansible Inventory file (inventory.txt)

```
1 [masters]
2 10.0.5.67
3 10.0.5.68
4 10.0.5.69
5 [slaves]
6 10.0.5.70
7 10.0.5.71
8 10.0.5.72
```

29.0.5 Instructions for Deployment

The following commands complete NIST Pedestrian and Face Detection deployment on OpenStack.

Cloning Pedestrian Detection Repository from Github

Roles are included as submodules which require `--recursive` option to checkout them all.

```
1 !git clone --recursive https://github.com/futuresystems/pedestrian-and-face↔
   ↪ -detection.git
```

Change the following variable with actual ip addresses:

```
1 sample_inventory="" [masters]
2 10.0.5.67
3 10.0.5.68
4 10.0.5.69
5 [slaves]
6 10.0.5.70
7 10.0.5.71
8 10.0.5.72""
```

Create a `inventory.txt` file with the variable in your local directory.

```
1 !printf "$sample_inventory" > inventory.txt
2 !cat inventory.txt
```

Add `ansible.cfg` file with options for ssh host key checking and login name.

```
1 ansible_config="" [defaults]
2 host_key_checking=false
3 remote_user=ubuntu""
4 !printf "$ansible_config" > ansible.cfg
5 !cat ansible.cfg
```

Check accessibility by ansible ping like:

```
1 !ansible -m ping -i inventory.txt all
```

Make sure that you have a correct ssh key in your account otherwise you may encounter 'FAILURE' in the ping test above.

Ansible Playbook

We use a main ansible playbook to deploy software packages for NIST Pedestrian and Face detection which includes: - mesos - spark - zookeeper - opencv - INRIA Person dataset - Python script for the detection

```
1 !cd pedestrian-and-face-detection/ && ansible-playbook -i ../inventory.txt <-
  ↪ site.yml
```

The installation may take 30 minutes or an hour to complete.

29.0.6 OpenCV in Python

Before we run our code for this project, let's try OpenCV first to see how it works.

Import cv2

Let's import opencv python module and we will use images from the online database image-net.org to test OpenCV image recognition.

```
1 import cv2
```

Let's download a mailbox image with a red color to see if opencv identifies the shape with a color. The example file in this tutorial is:

```
1 !curl http://farm4.static.flickr.com/3061/2739199963_ee78af76ef.jpg > <-
  ↪ mailbox.jpg
```

```
100 167k 100 167k 0 0 686k 0 --:--:-- --:--:-- --:--:-- 684k
```

```
1 %matplotlib inline
```

```
1 from IPython.display import Image
2 mailbox_image = "mailbox.jpg"
3 Image(filename=mailbox_image)
```



You can try other images. Check out the image-net.org for mailbox images: <http://image-net.org/synset?wnid=n03710193>

Image Detection

Just for a test, let's try to detect a red color shaped mailbox using opencv python functions.

There are key functions that we use: * cvtColor: to convert a color space of an image * inRange: to detect a mailbox based on the range of red color pixel values * np.array: to define the range of red

color using a Numpy library for better calculation * findContours: to find a outline of the object * bitwise_and: to black-out the area of contours found

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # imread for loading an image
5 img = cv2.imread(mailbox_image)
6 # cvtColor for color conversion
7 hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
8
9 # define range of red color in hsv
10 lower_red1 = np.array([0, 50, 50])
11 upper_red1 = np.array([10, 255, 255])
12 lower_red2 = np.array([170, 50, 50])
13 upper_red2 = np.array([180, 255, 255])
14
15 # threshold the hsv image to get only red colors
16 mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
17 mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
18 mask = mask1 + mask2
19
20 # find a red color mailbox from the image
21 im2, contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.C↳
↳ CHAIN_APPROX_SIMPLE)
22
23 # bitwise_and to remove other areas in the image except the detected object
24 res = cv2.bitwise_and(img, img, mask = mask)
25
26 # turn off - x, y axis bar
27 plt.axis("off")
28 # text for the masked image
29 cv2.putText(res, "masked image", (20,300), cv2.FONT_HERSHEY_SIMPLEX, 2, ↳
↳ (255,255,255))
30 # display
31 plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
32 plt.show()

```



The red color mailbox is left alone in the image which we wanted to find in this example by opencv functions. You can try other images with different colors to detect the different shape of objects using findContours and inRange from opencv.

For more information, see the useful links below.

- contours features: http://docs.opencv.org/3.1.0/dd/d49/tutorial/_py/_contour/_features.html
- contours: http://docs.opencv.org/3.1.0/d4/d73/tutorial/_py/_contours/_begin.html
- red color in hsv: <http://stackoverflow.com/questions/30331944/finding-red-color-using-python-opencv>
- inrange: http://docs.opencv.org/master/da/d97/tutorial/_threshold/_inRange.html
- inrange: http://docs.opencv.org/3.0-beta/doc/py/_tutorials/py/_imgproc/py/_colorspaces/py/_colorspaces.html
- numpy: http://docs.opencv.org/3.0-beta/doc/py/_tutorials/py/_core/py/_basic/_ops/py/_basic/_ops.html

29.0.7 Human and Face Detection in OpenCV

INRIA Person Dataset

We use INRIA Person dataset to detect upright people and faces in images in this example. Let's download it first.

```
1 !curl ftp://ftp.inrialpes.fr/pub/lear/douze/data/INRIAPerson.tar > ↵
   ↵ INRIAPerson.tar

100 969M 100 969M 0 0 8480k 0 0:01:57 0:01:57 --:--:-- 12.4M

1 !tar xvf INRIAPerson.tar > logfile && tail logfile
```

Face Detection using Haar Cascades

This section is prepared based on the opencv-python tutorial: http://docs.opencv.org/3.1.0/d7/d8b/tutorial/_py/_face/_detection.html#gsc.tab=0

There is a pre-trained classifier for face detection, download it from here:

```
1 !curl https://raw.githubusercontent.com/opencv/opencv/master/data/↵
   ↵ haarcascades/haarcascade_frontalface_default.xml > ↵
   ↵ haarcascade_frontalface_default.xml

100 908k 100 908k 0 0 2225k 0 --:--:-- --:--:-- --:--:-- 2259k
```

This classifier XML file will be used to detect faces in images. If you like to create a new classifier, find out more information about training from here: http://docs.opencv.org/3.1.0/dc/d88/tutorial/_traincascade.html

Face Detection Python Code Snippet

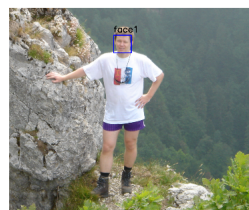
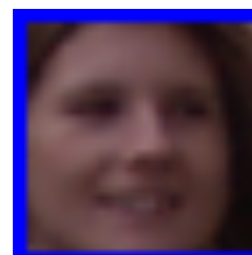
Now, we detect faces from the first five images using the classifier.

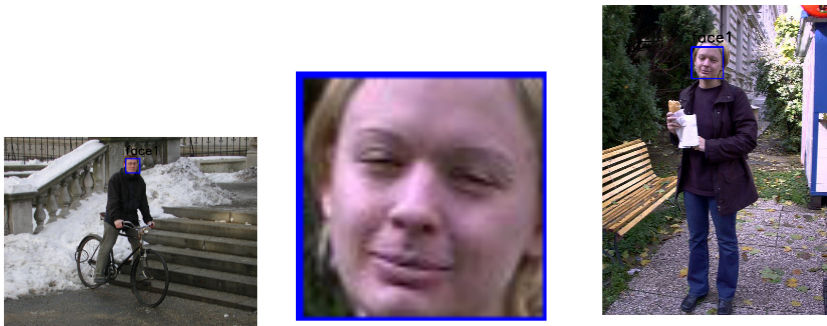
```
1 # import the necessary packages
2 from __future__ import print_function
3 import numpy as np
4 import cv2
5 from os import listdir
6 from os.path import isfile, join
7 import matplotlib.pyplot as plt
8
```

```

9 mypath = "INRIAPerson/Test/pos/"
10 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
11
12 onlyfiles = [join(mypath, f) for f in listdir(mypath) if isfile(join(mypath, f))]
13
14 cnt = 0
15 for filename in onlyfiles:
16     image = cv2.imread(filename)
17     image_grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
18     faces = face_cascade.detectMultiScale(image_grayscale, 1.3, 5)
19     if len(faces) == 0:
20         continue
21
22     cnt_faces = 1
23     for (x,y,w,h) in faces:
24         cv2.rectangle(image, (x,y), (x+w,y+h), (255,0,0), 2)
25         cv2.putText(image, "face" + str(cnt_faces), (x,y-10), cv2.↵
↵ FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
26         plt.figure()
27         plt.axis("off")
28         plt.imshow(cv2.cvtColor(image[y:y+h, x:x+w], cv2.COLOR_BGR2RGB))
29         cnt_faces += 1
30     plt.figure()
31     plt.axis("off")
32     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
33     cnt = cnt + 1
34     if cnt == 5:
35         break

```





29.0.8 Pedestrian Detection using HOG Descriptor

We will use Histogram of Oriented Gradients (HOG) to detect a upright person from images.

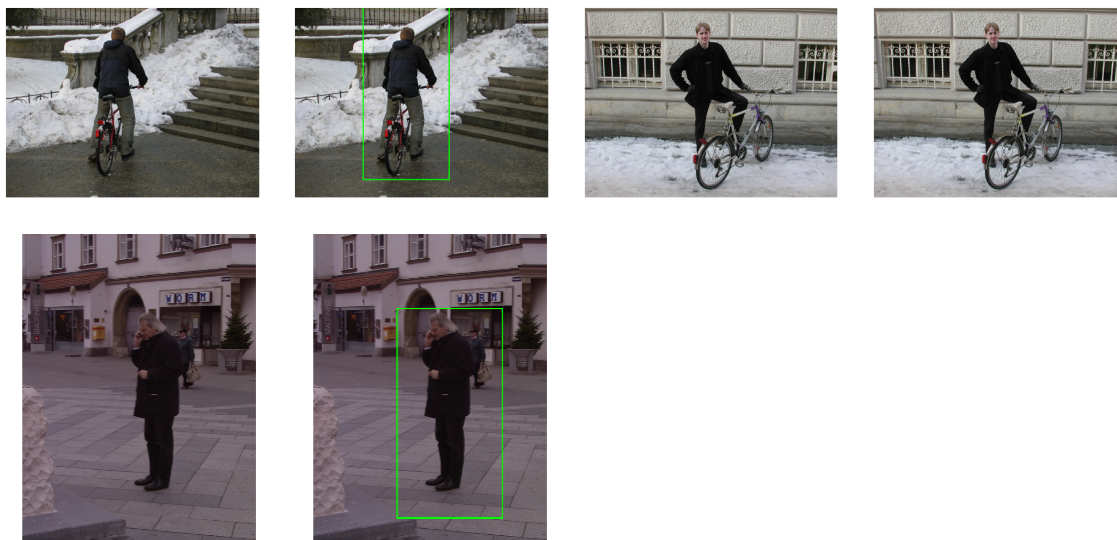
Python Code Snippet

```

1 # initialize the HOG descriptor/person detector
2 hog = cv2.HOGDescriptor()
3 hog.setSVMdetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
4
5 cnt = 0
6 for filename in onlyfiles:
7     img = cv2.imread(filename)
8     orig = img.copy()
9     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
10
11     # detect people in the image
12     (rects, weights) = hog.detectMultiScale(img, winStride=(8, 8),
13     padding=(16, 16), scale=1.05)
14
15     # draw the final bounding boxes
16     for (x, y, w, h) in rects:
17         cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
18
19     plt.figure()
20     plt.axis("off")
21     plt.imshow(cv2.cvtColor(orig, cv2.COLOR_BGR2RGB))
22     plt.figure()
23     plt.axis("off")
24     plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
25     cnt = cnt + 1
26     if cnt == 5:
27         break

```





29.0.9 Processing by Apache Spark

INRIA Person dataset provides 100+ images and Spark can be used for image processing in parallel. We load 288 images from ‘‘Test/pos’’ directory.

Spark provides a special object ‘sc’ to connect between a spark cluster and functions in python code. Therefore, we can run python functions in parallel to detect objects in this example.

- *map* function is used to process pedestrian and face detection per image from the `parallelize()` function of ‘sc’ spark context.
- *collect* function merges results in an array.

```

1 def apply_batch(imagePath):
2     import cv2
3     import numpy as np
4     # initialize the HOG descriptor/person detector
5     hog = cv2.HOGDescriptor()
6     hog.setSVMdetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
7     image = cv2.imread(imagePath)
8     # detect people in the image
9     (rects, weights) = hog.detectMultiScale(image, winStride=(8, 8),
10      padding=(16, 16), scale=1.05)
11     # draw the final bounding boxes
12     for (x, y, w, h) in rects:
13         cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
14     return image

```

Parallelize in Spark Context

The list of image files is given to parallelize.

```

1 pd = sc.parallelize(onlyfiles)

```

Map Function (apply_batch)

The ‘`apply_batch`’ function that we created above is given to map function to process in a spark cluster.


```
1 pdc = pd.map(apply_batch)
```

Collect Function

The result of each map process is merged to an array.

```
1 result = pdc.collect()
```

29.0.10 Results for 100+ images by Spark Cluster

```
1 for image in result:  
2     plt.figure()  
3     plt.axis("off")  
4     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```


30. Python Fingerprint Example

Python is a flexible and popular language for running data analysis pipelines. In this tutorial we will implement a solution for a fingerprint matching.

30.1 Overview

Fingerprint recognition refers to the automated method for verifying a match between two fingerprints and that is used to identify individuals and verify their identity. Fingerprints (Figure 30.1) are the most widely used form of biometric used to identify individuals.

TODO: image missing



Figure 30.1: Fingerprints

The automated fingerprint matching generally required the detection of different fingerprint features (aggregate characteristics of ridges, and minutia points) and then the use of fingerprint matching algorithm, which can do both one-to- one and one-to- many matching operations. Based on the number of matches a proximity score (distance or similarity) can be calculated.

We use the following NIST dataset for the study:

Special Database 14 - NIST Mated Fingerprint Card Pairs 2. (<http://www.nist.gov/itl/iad/>

[ig/special/_dbases.cfm](#))

TODO: citation

30.2 Objectives

Match the fingerprint images from a probe set to a gallery set and report the match scores.

30.3 Prerequisites

For this work we will use the following algorithms:

- MINDTCT: The NIST minutiae detector, which automatically locates and records ridge ending and bifurcations in a fingerprint image. (<http://www.nist.gov/itl/iad/ig/nbis.cfm>)
- BOZORTH3: A NIST fingerprint matching algorithm, which is a minutiae based fingerprint-matching algorithm. It can do both one-to-one and one-to-many matching operations. (<http://www.nist.gov/itl/iad/ig/nbis.cfm>)

In order to follow along, you must have the NBIS tools which provide `mindtct` and `bozorth3` installed. If you are on Ubuntu 16.04 Xenial, the following steps will accomplish this:

```
1 $ sudo apt-get update -qq
2 $ sudo apt-get install -y build-essential cmake unzip
3 $ wget "http://nigos.nist.gov:8080/nist/nbis/nbis_v5_0_0.zip"
4 $ unzip -d nbis nbis_v5_0_0.zip
5 $ cd nbis/Rel_5.0.0
6 $ ./setup.sh /usr/local --without-X11
7 $ sudo make
```

30.4 Implementation

1. Fetch the fingerprint images from the web
2. Call out to external programs to prepare and compute the match scores
3. Store the results in a database
4. Generate a plot to identify likely matches.

```
1 from __future__ import print_function
2 import urllib
3 import zipfile
4 import hashlib
```

We'll be interacting with the operating system and manipulating files and their pathnames.

```
1 import os.path
2 import os
3 import sys
4 import shutil
5 import tempfile
```

Some general useful utilities

```

1 import itertools
2 import functools
3 import types
4 from pprint import pprint

```

Using the `attrs` library provides some nice shortcuts to defining objects

```
1 import attr
```

```
1 import sys
```

We'll be randomly dividing the entire dataset, based on user input, into the probe and gallery sets

```
1 import random
```

We'll need to call out to the NBIS software. We'll also be using multiple processes to take advantage of all the cores on our machine

```

1 import subprocess
2 import multiprocessing

```

As for plotting, we'll use `matplotlib`, though there are many alternatives.

```

1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import numpy as np

```

Finally, we'll write the results to a database.

```
1 import sqlite3
```

30.5 Utility functions

Next, we'll define some utility functions:

```

1 def take(n, iterable):
2     "Returns a generator of the first **n** elements of an iterable"
3     return itertools.islice(iterable, n )
4
5
6 def zipWith(function, *iterables):
7     "Zip a set of **iterables** together and apply **function** to each ↔
8     ↔ tuple"
9     for group in itertools.izip(*iterables):
10         yield function(*group)
11
12 def uncurry(function):
13     "Transforms an N-ary **function** so that it accepts a single ↔
14     ↔ parameter of an N-tuple"
15     @functools.wraps(function)
16     def wrapper(args):
17         return function(*args)
18     return wrapper
19

```

```

20 def fetch_url(url, sha256, prefix='.', checksum_blocksize=2**20, dryRun=False):
21     """Download a url.
22
23     :param url: the url to the file on the web
24     :param sha256: the SHA-256 checksum. Used to determine if the file was
25     ↪ previously downloaded.
26     :param prefix: directory to save the file
27     :param checksum_blocksize: blocksize to used when computing the
28     ↪ checksum
29     :param dryRun: boolean indicating that calling this function should do
30     ↪ nothing
31     :returns: the local path to the downloaded file
32     :rtype:
33
34     """
35
36     if not os.path.exists(prefix):
37         os.makedirs(prefix)
38
39     local = os.path.join(prefix, os.path.basename(url))
40
41     if dryRun: return local
42
43     if os.path.exists(local):
44         print ('Verifying checksum')
45         chk = hashlib.sha256()
46         with open(local, 'rb') as fd:
47             while True:
48                 bits = fd.read(checksum_blocksize)
49                 if not bits: break
50                 chk.update(bits)
51             if sha256 == chk.hexdigest():
52                 return local
53
54     print ('Downloading', url)
55
56     def report(sofar, blocksize, totalsize):
57         msg = '{}\r'.format(100 * sofar * blocksize / totalsize, 100)
58         sys.stderr.write(msg)
59
60     urllib.urlretrieve(url, local, report)
61
62     return local

```

30.6 Dataset

We'll now define some global parameters

First, the fingerprint dataset

```

1 DATASET_URL = 'https://s3.amazonaws.com/nist-srd/SD4/
2 ↪ NISTSpecialDatabase4GrayScaleImagesofFIGS.zip'
3
4 DATASET_SHA256 = '4
5 ↪ db6a8f3f9dc14c504180cbf67cdf35167a109280f121c901be37a80ac13c449'

```


We'll define how to download the dataset. This function is general enough that it could be used to retrieve most files, but we'll default it to use the values from above.

```

1 def prepare_dataset(url=None, sha256=None, prefix='.', skip=False):
2     url = url or DATASET_URL
3     sha256 = sha256 or DATASET_SHA256
4     local = fetch_url(url, sha256=sha256, prefix=prefix, dryRun=skip)
5
6     if not skip:
7         print('Extracting', local, 'to', prefix)
8         with zipfile.ZipFile(local, 'r') as zip:
9             zip.extractall(prefix)
10
11     name, _ = os.path.splitext(local)
12     return name
13
14
15 def locate_paths(path_md5list, prefix):
16     with open(path_md5list) as fd:
17         for line in itertools.imap(str.strip, fd):
18             parts = line.split()
19             if not len(parts) == 2: continue
20             md5sum, path = parts
21             chksum = Checksum(value=md5sum, kind='md5')
22             filepath = os.path.join(prefix, path)
23             yield Path(checksum=chksum, filepath=filepath)
24
25
26 def locate_images(paths):
27
28     def predicate(path):
29         _, ext = os.path.splitext(path.filepath)
30         return ext in ['.png']
31
32     for path in itertools.ifilter(predicate, paths):
33         yield image(id=path.checksum.value, path=path)

```

30.7 Data Model

We'll define some classes so we have a nice API for working with the dataflow. We set `slots=True` so that the resulting objects will be more space-efficient.

30.7.1 Utilities

30.7.2 Checksum

The checksum consists of the actual hash value (`value`) as well as a string representing the hashing algorithm. The validator enforces that the algorithm can only be one of the listed acceptable methods

```

1 @attr.s(slots=True)
2 class Checksum(object):
3     value = attr.ib()
4     kind = attr.ib(validator=lambda o, a, v: v in 'md5 sha1 sha224 sha256 ↵
    ↵ sha384 sha512'.split())

```

30.7.3 Path

Paths refer to an image's filepath and associated Checksum. We get the checksum “for” free since the MD5 hash is provided for each image in the dataset.

```

1 @attr.s(slots=True)
2 class Path(object):
3     checksum = attr.ib()
4     filepath = attr.ib()

```

30.7.4 Image

The start of the data pipeline is the image. An image has an id (the md5 hash) and the path to the image.

```

1 @attr.s(slots=True)
2 class image(object):
3     id = attr.ib()
4     path = attr.ib()

```

30.7.5 Mindtct

The next step in the pipeline is to apply the mindtct program from NBIS. A mindtct object therefore represents the results of applying mindtct on an image. The xyt output is needed for the next step, and the image attribute represents the image id.

```

1 @attr.s(slots=True)
2 class mindtct(object):
3     image = attr.ib()
4     xyt = attr.ib()
5
6     def pretty(self):
7         d = dict(id=self.image.id, path=self.image.path)
8         return pprint(d)

```

We need a way to construct a mindtct object from an image object. A straightforward way of doing this would be to have a from_image @staticmethod or @classmethod, but that doesn't work well with multiprocessing as top-level functions work best as they need to be serialized.

```

1 def mindtct_from_image(image):
2     imgpath = os.path.abspath(image.path.filepath)
3     tempdir = tempfile.mkdtemp()
4     oroot = os.path.join(tempdir, 'result')
5
6     cmd = ['mindtct', imgpath, oroot]
7
8     try:
9         subprocess.check_call(cmd)
10
11         with open(oroot + '.xyt') as fd:
12             xyt = fd.read()
13
14         result = mindtct(image=image.id, xyt=xyt)
15         return result
16

```

```

17     finally:
18         shutil.rmtree(tempdir)

```

30.7.6 Bozorth3

The final step in the pipeline is running the bozorth3 from NBIS. The bozorth3 class represents the match being done: tracking the ids of the probe and gallery images as well as the match score.

Since we'll be writing these instance out to a database, we provide some static methods for SQL statements. While there are many Object-Relational-Model (ORM) libraries available for Python, this approach keeps the current implementation simple.

```

1  @attr.s(slots=True)
2  class bozorth3(object):
3      probe = attr.ib()
4      gallery = attr.ib()
5      score = attr.ib()
6
7      @staticmethod
8      def sql_stmt_create_table():
9          return 'CREATE TABLE IF NOT EXISTS bozorth3' \
10             + '(probe TEXT, gallery TEXT, score NUMERIC)'
11
12     @staticmethod
13     def sql_prepared_stmt_insert():
14         return 'INSERT INTO bozorth3 VALUES (?, ?, ?)'
15
16     def sql_prepared_stmt_insert_values(self):
17         return self.probe, self.gallery, self.score

```

In order to work well with multiprocessing, we define a class representing the input parameters to bozorth3 and a helper function to run bozorth3. This way the pipeline definition can be kept simple to a map to create the input and then a map to run the program.

As NBIS bozorth3 can be called to compare one-to-one or one-to-many, we'll also dynamically choose between these approaches depending on if the gallery attribute is a list or a single object.

```

1  @attr.s(slots=True)
2  class bozorth3_input(object):
3      probe = attr.ib()
4      gallery = attr.ib()
5
6      def run(self):
7          if isinstance(self.gallery, mindtct):
8              return bozorth3_from_one_to_one(self.probe, self.gallery)
9          elif isinstance(self.gallery, types.ListType):
10             return bozorth3_from_one_to_many(self.probe, self.gallery)
11          else:
12             raise ValueError('Unhandled type for gallery: {}'.format(type(←
← gallery)))

```

The next is the top-level function to running bozorth3. It accepts an instance of bozorth3_input. The is implemented as a simple top-level wrapper so that it can be easily passed to the multiprocessing library.

```

1  def run_bozorth3(input):

```

```
2     return input.run()
```

30.7.7 Running Bozorth3

There are two cases to handle: 1. One-to-one probe to gallery sets 1. One-to-many probe to gallery sets

Both approaches are implemented below. The implementations follow the same pattern: 1. Create a temporary directory within with to work 1. Write the probe and gallery images to files in the temporary directory 1. Call the bozorth3 executable 1. The match score is written to stdout which is captured and then parsed. 1. Return a bozorth3 instance for each match 1. Make sure to clean up the temporary directory

One-to-one

```
1 def bozorth3_from_one_to_one(probe, gallery):
2     tmpdir = tempfile.mkdtemp()
3     probeFile = os.path.join(tmpdir, 'probe.xyt')
4     galleryFile = os.path.join(tmpdir, 'gallery.xyt')
5
6     with open(probeFile, 'wb') as fd: fd.write(probe.xyt)
7     with open(galleryFile, 'wb') as fd: fd.write(gallery.xyt)
8
9     cmd = ['bozorth3', probeFile, galleryFile]
10
11     try:
12         result = subprocess.check_output(cmd)
13         score = int(result.strip())
14         return bozorth3(probe=probe.image, gallery=gallery.image, score=↔
↔ score)
15     finally:
16         shutil.rmtree(tmpdir)
```

One-to-many

```
1 def bozorth3_from_one_to_many(probe, galleryset):
2     tmpdir = tempfile.mkdtemp()
3     probeFile = os.path.join(tmpdir, 'probe.xyt')
4     galleryFiles = [os.path.join(tmpdir, 'gallery%d.xyt' % i)
5                     for i,_ in enumerate(galleryset)]
6
7     with open(probeFile, 'wb') as fd: fd.write(probe.xyt)
8     for galleryFile, gallery in itertools.izip(galleryFiles, galleryset):
9         with open(galleryFile, 'wb') as fd: fd.write(gallery.xyt)
10
11     cmd = ['bozorth3', '-p', probeFile] + galleryFiles
12
13     try:
14         result = subprocess.check_output(cmd).strip()
15         scores = map(int, result.split('\n'))
16         return [bozorth3(probe=probe.image, gallery=gallery.image, score=↔
↔ score)
17                 for score, gallery in zip(scores, galleryset)]
18     finally:
19         shutil.rmtree(tmpdir)
```


30.8 Plotting

For plotting we'll operate only on the database. We'll select a small number of probe images and plot the score between them and the rest of the gallery images.

The `mk_short_labels` helper function will be defined below.

```

1 def plot(dbfile, nprobes=10):
2     conn = sqlite3.connect(dbfile)
3     results = pd.read_sql(
4         "SELECT DISTINCT probe FROM bozorth3 ORDER BY score LIMIT '%s'" % ←
5         ↪ nprobes,
6         con=conn
7     )
8     shortlabels = mk_short_labels(results.probe)
9     plt.figure()
10
11     for i, probe in results.probe.iteritems():
12         stmt = 'SELECT gallery, score FROM bozorth3 WHERE probe = ? ORDER ←
13         ↪ BY gallery DESC'
14         matches = pd.read_sql(stmt, params=(probe,), con=conn)
15         xs = np.arange(len(matches), dtype=np.int)
16         plt.plot(xs, matches.score, label='probe %s' % shortlabels[i])
17
18     plt.ylabel('Score')
19     plt.xlabel('Gallery')
20     plt.legend(bbox_to_anchor=(0, 0, 1, -0.2))
21     plt.show()

```

The image ids are long hash strings. In order to minimize the amount of space on the figure the labels occupy, we provide a helper function to create a short label that still uniquely identifies each probe image in the selected sample

```

1 def mk_short_labels(series, start=7):
2     for size in xrange(start, len(series[0])):
3         if len(series) == len(set(map(lambda s: s[:size], series))):
4             break
5     return map(lambda s: s[:size], series)

```

30.9 Putting it all Together

First, set up a temporary directory in which to work:

```

1 pool = multiprocessing.Pool()
2 prefix = '/tmp/fingerprint_example/'
3 if not os.path.exists(prefix):
4     os.makedirs(prefix)

```

Next we download and extract the fingerprint images from NIST:

```

1 dataprefix = prepare_dataset(prefix=prefix)

```

Next we'll configure the location of the MD5 checksum file that comes with the download

```

1 md5listpath = os.path.join(prefix, '←
2     ↪ NISTSpecialDatabase4GrayScaleImagesofFIGS/sd04/sd04_md5.lst')

```

Load the images from the downloaded files to start the analysis pipeline

```
1 print('Loading images')
2 paths = locate_paths(md5listpath, dataprefix)
3 images = locate_images(paths)
4 mindtcts = pool.map(mindtct_from_image, images)
5 print('Done')
```

We can examine one of the loaded image. Note that image is refers to the MD5 checksum that came with the image and the xyt attribute represents the raw image data.

```
1 print(mindtcts[0].image)
2 print(mindtcts[0].xyt[:50])
```

For example purposes we'll only use a small percentage of the database, randomly selected, for our probe and gallery datasets.

```
1 perc_probe = 0.001
2 perc_gallery = 0.1

1 print('Generating samples')
2 probes = random.sample(mindtcts, int(perc_probe * len(mindtcts)))
3 gallery = random.sample(mindtcts, int(perc_gallery * len(mindtcts)))
4 print('|Probes| =', len(probes))
5 print('|Gallery|=', len(gallery))
```

We can now compute the matching scores between the probe and gallery sets. This will use all cores available on this workstation.

```
1 print('Matching')
2 input = [bozorth3_input(probe=probe, gallery=gallery)
3          for probe in probes]
4 bozorth3s = pool.map(run_bozorth3, input)
```

bozorth3s is now a list of lists of bozorth3 instances.

```
1 print('|Probes| =', len(bozorth3s))
2 print('|Gallery| =', len(bozorth3s[0]))
3 print('Result:', bozorth3s[0][0])
```

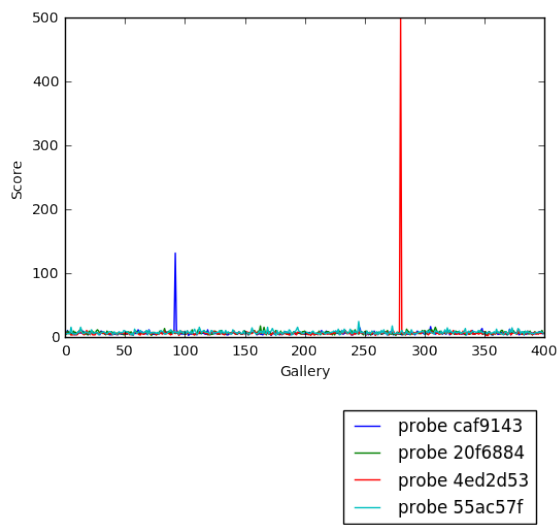
Now add the results to the database

```
1 dbfile = os.path.join(prefix, 'scores.db')
2 conn = sqlite3.connect(dbfile)
3 cursor = conn.cursor()
4 cursor.execute(bozorth3.sql_stmt_create_table())

1 for group in bozorth3s:
2     vals = map(bozorth3.sql_prepared_stmt_insert_values, group)
3     cursor.executemany(bozorth3.sql_prepared_stmt_insert(), vals)
4     conn.commit()
5     print('Inserted results for probe', group[0].probe)
```

We now plot the results.

```
1 plot(dbfile, nprobes=len(probes))
```



```
1 cursor.close()
```




Cloudmesh

31	Cloudmesh Command Shell	387
31.1	CMD5	
32	Draft: Enhanced Cloudmesh	391
32.1	Configuration	
32.2	Storage	



31. Cloudmesh Command Shell

31.1 CMD5

Python's CMD (<https://docs.python.org/2/library/cmd.html>) is a very useful package to create command line shells. However it does not allow the dynamic integration of newly defined commands. Furthermore, additions to CMD need to be done within the same source tree. To simplify developing commands by a number of people and to have a dynamic plugin mechanism, we developed cmd5. It is a rewrite on our earlier efforts in cloudmesh client and cmd3.

31.1.1 Resources

The source code for cmd5 is located in github:

- <https://github.com/cloudmesh/cmd5>

31.1.2 Creating a Python Development Environment

We recommend that you use a virtualenv either with virtualenv or pyenv. This is in detail documented in the Section [22.1.1](#).

31.1.3 Installation from source

Cmd5 can be easily deployed with pip:

```
pip install cloudmesh.cmd5
```

In case you would like to generate easily new cmd5 commands we also recommend you install the cloudmesh sys command with:

```
pip install cloudmesh.sys
```

In case you like to work with the source please clone the following directories from github:

```
mkdir -p ~/github
cd ~/github

git clone https://github.com/cloudmesh/cloudmesh.common.git
git clone https://github.com/cloudmesh/cloudmesh.cmd5.git
git clone https://github.com/cloudmesh/cloudmesh.sys.git

cd ~/github/cloudmesh.common
python setup.py install
pip install .

cd ~/github/cloudmesh.cmd5
python setup.py install
pip install .

cd ~/github/cloudmesh.sys
python setup.py install
pip install .
```

The common directory contains some useful libraries, the cmd5 repository contains the shell, while the sys directory contains a command to generate extensions to cloudmesh.

31.1.4 Execution

To run the shell you can activate it with the cms command. cms stands for cloudmesh shell:

```
(ENV2) $ cms
```

It will print the banner and enter the shell:

```
+-----+
| / _ _ _ | | _ _ _ | | _ _ _ | | _ _ _ | | _ _ _ | | _ _ _ | | | | | | | | | | | | | | | | | |
| | | | | | / _ \ | | / _ \ | | ' ' _ \ / _ \ | | ' _ \ |
| | | | | | ( ) | | | | ( | | | | | | | | | | | | | | | | | | |
| \ _ _ _ / \ _ _ / \ _ _ \ | | | | | | \ _ _ / | | | | |
+-----+
|                               Cloudmesh CMD5 Shell                               |
+-----+

cms>
```

To see the list of commands you can say:

```
cms> help
```

To see the manual page for a specific command, please use:

```
help COMMANDNAME
```

31.1.5 Create your own Extension

One of the most important features of CMD5 is its ability to extend it with new commands. This is done via packaged name spaces. We recommend you name is cloudmesh.mycommand, where mycommand is the name of the command that you like to create. This can easily be done while using the sys command:

```
cms sys command generate mycommand
```

It will download a template from cloudmesh called cloudmesh.bar and generate a new directory cloudmesh.mycommand with all the needed files to create your own command and register it dynamically with cloudmesh. All you have to do is to cd into the directory and install the code:

```
cd cloudmesh.mycommand
python setup.py install
pip install .
```

Adding your own command is easy. It is important that all objects are defined in the command itself and that no global variables be use in order to allow each shell command to stand alone. Naturally you should develop API libraries outside of the cloudmesh shell command and reuse them in order to keep the command code as small as possible. We place the command in:

```
cloudmesh/mycommand/command/mycommand.py
```

An example for the bar command is presented at:

- <https://github.com/cloudmesh/cloudmesh.bar/blob/master/cloudmesh/bar/command/bar.py>

It shows how simple the command definition is (bar.py):

```
from __future__ import print_function
from cloudmesh.shell.command import command
from cloudmesh.shell.command import PluginCommand

class BarCommand(PluginCommand):

    @command
    def do_bar(self, args, arguments):
        """
        ::
        Usage:
            command -f FILE
            command FILE
            command list
        This command does some useful things.
        Arguments:
            FILE  a file name
        Options:
            -f    specify the file
        """
        print(arguments)
```

An important difference to other CMD solutions is that our commands can leverage (besides the standard definition), docopts as a way to define the manual page. This allows us to use arguments as dict and use simple if conditions to interpret the command. Using docopts has the advantage that contributors are forced to think about the command and its options and document them from the start. Previously we did not use but argparse and click. However we noticed that for our contributors both systems lead to commands that were either not properly documented or the developers delivered ambiguous commands that resulted in confusion and wrong usage by subsequent users. Hence, we do recommend that you use docopts for documenting cmd5 commands. The transformation is enabled by the @command decorator that generates a manual page and creates a proper help message for the shell automatically. Thus there is no need to introduce a separate help method as would normally be needed in CMD while reducing the effort it takes to contribute new commands in a dynamic fashion.

31.1.6 Exercises

Exercise 31.1 Install cmd5 on your computer. ■

Exercise 31.2 Write a new command with your firstname as the command name. ■

Exercise 31.3 Write a new command and experiment with docopt syntax and argument interpretation of the dict with if conditions. ■

Exercise 31.4 If you have useful extensions that you like us to add by default, please work with us. ■

Exercise 31.5 At this time one needs to quote in some commands the " in the shell command line. Develop and test code that fixes this. ■



32. Draft: Enhanced Cloudmesh

In this chapter we will be using some advanced Python features to enhance Cloudmesh. Cloudmesh is supposed to easily manage multiple clouds. We will be explicitly using python 3 and do not worry about backwards compatibility. It is a reimplementaion of earlier versions of cloudmesh, including cloudmesh client.

We will be developing it as community so that new features can be integrated and loaded on demand while adding an extensible package management system based on python's shared namespace. To do so we will rely on cmd5 that includes a generate command to add new packages on demand. We will not use all features of cmd5.

We are trying to develop the following:

Configuration so that we can easily configure and add various clouds to our multi-cloud environment.

Database of virtual machines and clouds so that they can be managed across different clouds in a multi cloud environment

API Classes so that we can use python as a convenient programming environment.

Context libraries so that in python we can easily apply context for clouds and virtual machines on a block of statements

Command Shell so that we can similar to matlab and other shells execute multiple commands

REST Services so that we can access the features from other programming environments and different programming languages.

Parallel Services so that we can issue commands in parallel and manage virtual machines in a multi cloud environment.

32.1 Configuration

As we are developing a multi-cloud environment, we need some mechanism to define the clouds easily. To make our development effort simpler, we like to point out that the configuration file must be stored in a particular location relative to the home directory. We store the file in `~/.cloudmesh/class.yaml`. Additionally we store our cloud passwords in this file in cleartext and thus we must make sure our machine is not compromised and that we properly protect the file. On a unix system you do this with:

```
mkdir ~/.cloudmesh
touch ~/.cloudmesh/class.yaml
chmod go-rw ~/.cloudmesh
chmod go-rw ~/.cloudmesh/class.yaml
```

In that directory we store a file similar to the following file:

```
version: 5.0
profile:
  firstname: Gregor
  lastname: von Laszewski
  email: laszewski@gmail.com
cloudmesh:
  default:
    - chameleon
  active:
    - chameleon
clouds:
  uc:
    name: Chameleon UC
    host: chameleoncloud.org
    type: openstack
    version: liberty
    credentials:
      OS_AUTH_URL: https://openstack.uc.chameleoncloud.org:5000/v3
      OS_PASSWORD: TBD
      OS_TENANT_NAME: CH-818664
      OS_TENANT_ID: CH-818664
      OS_PROJECT_NAME: CH-818664
      OS_PROJECT_DOMAIN_ID: default
      OS_USER_DOMAIN_ID: default
      OS_USERNAME: TBD
      OS_VERSION: liberty
      OS_REGION_NAME: RegionOne
    default:
      flavor: m1.small
      image: Ubuntu-Server-14.04-LTS
  tacc:
    name: Chameleon TACC
    host: chameleoncloud.org
    type: openstack
    version: liberty
    credentials:
      OS_AUTH_URL: https://openstack.tacc.chameleoncloud.org:5000/v3
      OS_PASSWORD: TBD
      OS_TENANT_NAME: CH-818664
      OS_TENANT_ID: CH-818664
      OS_PROJECT_NAME: CH-818664
      OS_PROJECT_DOMAIN_ID: default
      OS_USER_DOMAIN_ID: default
      OS_USERNAME: TBD
      OS_VERSION: liberty
```

```

OS_REGION_NAME: RegionOne
default:
  flavor: m1.small
  image: Ubuntu-Server-14.04-LTS

```

Important to note is that this file defines multiple clouds and uses the attribute value TBD for password and username which you may want to change. However, we also would like to support a mode that when the password is defined to be TBD that it is asked from the terminal. This way we do not necessarily have to store the password here. In future we will enhance this file to be encrypted and decrypted with a password protected ssh key.

The file is a yaml file as the typical configuration in python is not suitable to easily store hierarchical data. YAML is also more readable than json so it provides a really good way of defining the configuration data. Problematic with yaml readers however are that they typically do not preserve the read order. Your task will be to write a *short* yaml configuration reader that preserves the order. You are encouraged to reuse methods. What you are not supposed to do is to reimplement yaml.

There are some special properties of this file that we need to discuss.

- clouds are listed in the clouds section
- the credentials section to each cloud defines how to connect to the cloud with python libraries such as libcloud. Each cloud type will have different parameters.
-

32.2 Storage

As we need to store some of the data we must identify a suitable database for storing information about virtual machines and other information related to the clouds. Although shelve comes in mind, we found out that it is not compatible between python 2 and 3 which may be an issue in future. Also when considering services such as mongodb they have to be started and properly secured. This naturally can be done with containers. We also do not want to use large frameworks such as django which come with build in object models as they are not lightweight. Hence, we start we just use a file based sql database as provided with sqlite3.

32.2.1 sqlite3

While we keep the configuration in the configuration yaml file we intend to create a database entry for virtual machines we start in the cloud. In order to store hierarchical information that we may obtain in dict format from a virtual machine we can easily create flattened out data structures, by simple connecting the attribute names and separate them by `_`.

Let us assume we want to store an object of the following form:

```

element = {
  'id': 1,
  'cloud': 'chameleon',
  'name': 'vm1',
  'data': {
    'image': 'ubuntu',
    'flavor': 'small'
  }
}

```

A table that could store such an object could be

```

create table element (

```

```

        id            integer primary key,
        name          text
        cloud         text,
        data_image:   text,
        data_flavor:  text
    );

```

Obviously, we could create the table automatically from recursively iterating through the dict to make our approach generalized for any dict. As for the primary key, we simply assume it is always the id which is an integer that always increases and is stored in the database. as a separate element.

Thus we probably want a table generator such as

```

248 class Database (object):
249     @staticmethod
250     def generate (dictionary):
251         # implement me

```

Additionally we want to create convenience methods for adding, deleting, and searching information

32.2.2 Context

Python provides the feature of a context that we are well familiar with from file management. An example is:

```

252 with open('/tmp/gregor.txt', 'wt') as f:
253     f.write('Hello Gregor')
254 # after this, the file is automatically closed

```

If we look at this example it is desirable to develop at least two context for multicloud environments. The first is to manage virtual machines on named clouds and issue action on it, such as *start*, *stop*, *suspend*, *resume*, *delete*. In the other context we like to issue such action on named virtual machines.

To illustrate what we have in mind, please take a look at our initial examples.

Cloud Context

When defining the following cloud context

```

255 class Cloud (object):
256
257     def __init__(self, name):
258         self.name = name
259
260     def __enter__(self):
261         print ('Running on:', self.name)
262         return self
263
264     def __exit__(self, exc_type, exc_val, exc_tb):
265         print ('__exit__()')
266
267     def machine(self, name, action):
268         print (name, action)

```

we can issue conveniently commands such as the following

```

269 cloud = 'chameleon'

```



```
270 with Cloud(cloud) as c:  
271     vm = c.machine('vm1', 'start')
```

It is obvious that through this abstraction we can formulate a templated behavior such as starting a virtual machine and through the switch of a single variable (`cloud`) issue the command on other clouds.



Cloud Computing

33	Overview	399
33.1	Introduction to Cloud Computing	



33. Overview

33.1 Introduction to Cloud Computing

This introduction to Cloud Computing covers all aspects of the field drawing on industry and academic advances. It makes use of analyses from the Gartner group on future Industry trends. The presentation is broken into 21 parts starting with a survey of all the material covered. Note this first part is A while the substance of the talk is in parts B to U.

33.1.1 Introduction - Part A

- Parts B to D **define cloud computing**, its key concepts and how it is situated in the data center space
- The next part E reviews **virtualization** technologies comparing containers and hypervisors
- Part F is the first on **Gartner's Hypecycles** and especially those for emerging technologies in 2017 and 2016
- Part G is the second on **Gartner's Hypecycles** with Emerging Technologies hypecycles and the Priority matrix at selected times 2008-2015
- Parts H and I cover **Cloud Infrastructure** with Comments on trends in the data center and its technologies and the Gartner hypecycle and priority matrix on Infrastructure Strategies and Compute Infrastructure
- Part J covers **Cloud Software** with HPC-ABDS(High Performance Computing enhanced Apache Big Data Stack) with over 350 software packages and how to use each of its 21 layers
- Part K is first on **Cloud Applications** covering those from industry and commercial usage patterns from NIST
- Part L is second on **Cloud Applications** covering those from science where area called cyberinfrastructure; we look at the science usage pattern from NIST
- Part M is third on **Cloud Applications** covering the characterization of applications using the NIST approach.
- Part N covers **Clouds and Parallel Computing** and compares Big Data and Simulations
- Part O covers **Cloud storage**: Cloud data approaches: Repositories, File Systems, Data lakes
- Part P covers **HPC and Clouds** with The Branscomb Pyramid and Supercomputers versus clouds
- Part Q compares **Data Analytics with Simulation** with application and software implications
- Part R compares **Jobs** from Computer Engineering, Clouds, Design and Data Science/Engineering
- Part S covers the **Future** with Gartner cloud computing hypecycle and priority matrix, Hyperscale computing, Serverless and FaaS, Cloud Native and Microservices
- Part T covers **Security and Blockchain**
- Part U covers **fault-tolerance**

[Introduction - Part A \(14:48\)](#) 

[Introduction - Part A \(7 Slides\)](#) 

This lecture describes the contents of the following 20 parts (B to U).

33.1.2 Introduction - Part B - Defining Clouds I

[Part B - Defining Clouds I \(20:22\)](#) 

[Part B - Defining Clouds I \(13 Slides\)](#) 

B: Defining Clouds I

- Basic definition of cloud and two very simple examples of why virtualization is important.
- How clouds are situated wrt HPC and supercomputers
- Why multicore chips are important
- Typical data center

33.1.3 Introduction - Part C - Defining Clouds II

[Part C - Defining Clouds II \(20:45\)](#) 

[Part C - Defining Clouds II \(11 Slides\)](#) 

C: Defining Clouds II

- Service-oriented architectures: Software services as Message-linked computing capabilities
- The different aaS's: Network, Infrastructure, Platform, Software
- The amazing services that Amazon AWS and Microsoft Azure have
- Initial Gartner comments on clouds (they are now the norm) and evolution of servers; serverless and microservices

33.1.4 Introduction - Part D - Defining Clouds III

[Part D - Defining Clouds III \(9:08\)](#) 

[Part D - Defining Clouds III \(9 Slides\)](#) 

D: Defining Clouds III

- Cloud Market Share
- How important are they?
- How much money do they make?

33.1.5 Introduction - Part E - Virtualization

[Part E - Virtualization \(11:21\)](#) 

[Part E - Virtualization \(8 Slides\)](#) 

E: Virtualization

- Virtualization Technologies, Hypervisors and the different approaches
- KVM Xen, Docker and Openstack
- Several web resources are listed

33.1.6 Introduction - Part F - Technology Hypecycle I

[Part F - Technology Hypecycle I \(13:41\)](#) 

[Part F - Technology Hypecycle I \(11 Slides\)](#) 

F:Technology Hypecycle I

- Gartner's Hypecycles and especially that for emerging technologies in 2017 and 2016
- The phases of hypecycles
- Priority Matrix with benefits and adoption time
- Today clouds have got through the cycle (they have emerged) but features like blockchain, serverless and machine learning are on cycle
- Hypecycle and Priority Matrix for Data Center Infrastructure 2017

33.1.7 Introduction - Part G - Technology Hypecycle II

[Part G - Technology Hypecycle II \(16:05\)](#) 

[Part G - Technology Hypecycle II \(15 Slides\)](#) 

G: Technology Hypecycle II

- Emerging Technologies hypecycles and Priority matrix at selected times 2008-2015
- Clouds star from 2008 to today
- They are mixed up with transformational and disruptive changes
- The route to Digital Business (2015)

33.1.8 Introduction - Part H - IaaS I

Part H - IaaS I (13:22) 

Part H - IaaS I (12 Slides) 

H: Cloud Infrastructure I

- Comments on trends in the data center and its technologies
- Clouds physically across the world
- Green computing and fraction of world's computing ecosystem in clouds

33.1.9 Introduction - Part I - IaaS II

Part I - IaaS II (13:13) 

Part I - IaaS II (11 Slides) 

I: Cloud Infrastructure II

- Gartner hypecycle and priority matrix on Infrastructure Strategies and Compute Infrastructure
- Containers compared to virtual machines
- The emergence of artificial intelligence as a dominant force

33.1.10 Introduction - Part J - Cloud Software

Part J - Cloud Software (37:56) 

Part J - Cloud Software (15 Slides) 

J: Cloud Software

- HPC-ABDS(High Performance Computing enhanced Apache Big Data Stack) with over 350 software packages and how to use each of 21 layers
- Google's software innovations
- MapReduce in pictures
- Cloud and HPC software stacks compared
- Components need to support cloud/distributed system programming
- Single Program/Instruction Multiple Data SIMD SPMD

33.1.11 Introduction - Part K - Applications I

Part K - Applications I (11:58) 

Part K - Applications I (16 Slides) 

K: Cloud Applications I

- Big Data in Industry/Social media; a lot of best examples have NOT been updated so some slides old but still make the correct points
- Some of the business usage patterns from NIST

33.1.12 Introduction - Part L - Applications II*Part L - Applications II (13:03)* *Part L - Applications II (11 Slides)* **L: Cloud Applications II**

- Clouds in science where area called cyberinfrastructure;
- The science usage pattern from NIST
- Artificial Intelligence from Gartner

33.1.13 Introduction - Part M - Applications III*Part M - Applications III (24:12)* *Part M - Applications III (14 Slides)* **M: Cloud Applications III**

- Characterize Applications using NIST approach
- Internet of Things
- Different types of MapReduce

33.1.14 Introduction - Part N - Parallelism*Part N - Parallelism (35:46)* *Part N - Parallelism (15 Slides)* **N: Clouds and Parallel Computing**

- Parallel Computing in general
- Big Data and Simulations Compared
- What is hard to do?

33.1.15 Introduction - Part O - Storage*Part O - Storage (19:22)* *Part O - Storage (10 Slides)* **O: Cloud Storage**

- Cloud data approaches
- Repositories, File Systems, Data lakes

33.1.16 Introduction - Part P - HPC in the Cloud*Part P - HPC in the Clou (19:29)* *Part P - HPC in the Clou (8 Slides)* **P: HPC and Clouds**

- The Branscomb Pyramid
- Supercomputers versus clouds
- Science Computing Environments

33.1.17 Introduction - Part Q - Analytics and Simulation*Part Q - Analytics and Simulation (16:19)*  *Part Q - Analytics and Simulation (10 Slides)* **Q: Comparison of Data Analytics with Simulation**

- Structure of different applications for simulations and Big Data
- Software implications
- Languages

33.1.18 Introduction - Part R - Jobs

Part R - Jobs (4:52) 

Part R - Jobs (6 Slides) 

R: Availability of Jobs in different areas

- Computer Engineering
- Clouds
- Design
- Data Science/Engineering

33.1.19 Introduction - Part S - The Future

Part S - The Future (19:46) 

Part S - The Future (6 Slides) 

S: The Future

- Gartner cloud computing hypecycle and priority matrix highlights:
 - Hyperscale computing
 - Serverless and FaaS
 - Cloud Native
 - Microservices

33.1.20 Introduction - Part T - Security

Part T - Security (11:29) 

Part T - Security (13 Slides) 

T: Security

- CIO Perspective
- Blockchain

33.1.21 Introduction - Part U - Fault Tolerance

Part U - Fault Tolerance (9:10) 

Part U - Fault Tolerance (5 Slides) 

U: Fault Tolerance

- S3 Fault Tolerance
- Application Requirements

XI Cloud Server Technologies


34	REST	407
34.1	Overview of REST	
34.2	Flask RESTful Services	
34.3	Rest Services with Eve	
34.4	Object Management with Eve and Evegenie	
34.5	Towards cmd5 extensions to manage eve and mongo	
34.6	Responses	
34.7	Django REST Framework	
34.8	Rest Services with Swagger	
34.9	Swagger Tools	
34.10	Swagger Community Tools	
34.11	REST Service Generation with Swagger	
34.12	Swagger Specification	
35	MQTT	439
35.1	Introduction	
35.2	Publish Subscribe Model	
35.3	Secure MQTT Services	
35.4	Integration with Other Services	
35.5	MQTT in Production	
35.6	Simple Usecase	
35.7	IoT Use Case	
35.8	Conclusion	
35.9	Exercises	
36	GraphQL	447



34. REST

34.1 Overview of REST

This section is accompanied by a video about REST.

[REST \(36:02\)](#) 

REST stands for **RE**presentational **S**tate **T**ransfer. REST is an architecture style for designing networked applications. It is based on stateless, client-server, cacheable communications protocol. In contrast to what some others write or say, REST is not a *standard*. Although not based on http, in most cases, the HTTP protocol is used. In that case, RESTful applications use HTTP requests to (a) post data while creating and/or updating it, (b) read data while making queries, and (c) delete data.

REST was first introduced in a thesis from Fielding:

- <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Hence REST can use HTTP for the four CRUD operations:

- **C**reate resources
- **R**ead resources
- **U**ppdate resources
- **D**elete resources

As part of the HTTP protocol we have methods such as GET, PUT, POST, and DELETE. These methods can then be used to implement a REST service. This is not surprising as the HTTP protocol was explicitly designed to support these operations. As REST introduces collections and items we need to implement the CRUD functions for them. We distinguish single resources and collection of resources. The semantics for accessing them is explained next illustrating how to imple-

[section/rest/rest.tex](#)

ment them with HTTP methods (see https://en.wikipedia.org/wiki/Representational_state_transfer).

34.1.1 Collection of Resources

Let us assume the following URI identifies a collection of resources

`http://.../resources/`

than we need to implement the following CRUD methods:

GET List the URIs and perhaps other details of the collections members

PUT Replace the entire collection with another collection.

POST Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.

DELETE Delete the entire collection.

34.1.2 Single Resource

Let us assume the following URI identifies a single resource in a collection of resources

`http://.../resources/item42`

than we need to implement the following CRUD methods:

GET Retrieve a representation of the addressed member of the collection, expressed in an appropriate internet media type.

PUT Replace the addressed member of the collection, or if it does not exist, create it.

POST Not generally used. Treat the addressed member as a collection in its own right and create a new entry within it.

DELETE Delete the addressed member of the collection.

34.1.3 REST Tool Classification

Due to the well defined structure that REST provides a number of tools have been created that manage the creation of the specification for rest services and their programming. We distinguish several different categories:

REST programming language support: These tools and services are targeting a particular programming language. Such tools include Eve which we will explore in more detail.

REST documentation based tools: These tools are primarily focussing on documenting REST specifications. Such tools include Swagger, which we will explore in more detail.

REST design support tools: These tools are used to support the design process of developing REST services while abstracting on top of the programming languages and define reusable specifications that can be used to create clients and servers for particular technology targets. Such tools include also swagger as additional tools are available that can generate code from swagger specifications, which we will explore in more detail.

34.2 Flask RESTful Services

Flask is a micro services framework allowing to write web services in python quickly. One of its extensions is Flask-RESTful. It adds for building REST APIs based on a class definition making it

relatively simple. Through tis interface we can than integrate with your existing Object Relational Modles and libraries. As Flask-RESTful leverages the main features from Flask an extensive set of documentation is available allowing you to get started quickly and thourghly. The Web page contains extensive documentation:

- <https://flask-restful.readthedocs.io/en/latest/>

We will provide a simple example that showcases some *hard coded* data to be served as a rest service. It will be easy to replace this for example with functions and methods that obtain such information dynamically from the operationg system.

Warning

This example has not been tested., We like that the E222 class defines a beautiful example to contribute to this section. and explains what happens in this example.

```
1 from flask import Flask
2 from flask_restful import reqparse, abort, Api, Resource
3
4 app = Flask(__name__)
5 api = Api(app)
6
7 COMPUTERS = {
8     'computer1': {
9         'processor': 'iCore7'
10    },
11    'computer2': {
12        'processor': 'iCore5'
13    },
14    'computer3': {
15        'processor': 'iCore3'
16    },
17 }
18
19 def abort_if_cluster_doesnt_exist(computer_id):
20     if computer_id not in COMPUTERS:
21         abort(404, message="Computer {} doesn't exist".format(computer_id))
22
23 parser = reqparse.RequestParser()
24 parser.add_argument('processor')
25
26 class Computer(Resource):
27     ''' shows a single computer item and lets you delete a computer
28         item.'''
29
30     def get(self, computer_id):
31         abort_if_computer_doesnt_exist(computer_id)
32         return COMPUTERS[computer_id]
33
34     def delete(self, computer_id):
35         abort_if_computer_doesnt_exist(computer_id)
36         del COMPUTERS[computer_id]
37         return '', 204
38
39     def put(self, computer_id):
40         args = parser.parse_args()
41         processor = {'processor': args['processor']}
```

```

42     COMPUTERS[computer_id] = processor
43     return processor, 201
44
45
46 # ComputerList
47 class ComputerList(Resource):
48     ''' shows a list of all computers, and lets you POST to add new ↔
49     ↔ computers'''
49
50     def get(self):
51         return COMPUTERS
52
53     def post(self):
54         args = parser.parse_args()
55         computer_id = int(max(COMPUTERS.keys()).rstrip('computer')) + 1
56         computer_id = 'computer%i' % computer_id
57         COMPUTERS[computer_id] = {'processor': args['processor']}
58         return COMPUTERS[computer_id], 201
59
60 ##
61 ## Setup the Api resource routing here
62 ##
63 api.add_resource(ComputerList, '/computers')
64 api.add_resource(Computer, '/computers/<computer_id>')
65
66
67 if __name__ == '__main__':
68     app.run(debug=True)

```

34.3 Rest Services with Eve

Next, we will focus on how to make a RESTful web service with Python Eve. Eve makes the creation of a REST implementation in python easy. More information about Eve can be found at:

- <http://python-eve.org/>

Warning

Although we do recommend Ubuntu 17.04, at this time there is a bug that forces us to use 16.04. Furthermore, we require you to follow the instructions on how to install pyenv and use it to set up your python environment. We recommend that you use either python 2.7.14 or 3.6.4. We do not recommend you to use anaconda as it is not suited for cloud computing but targets desktop computing. If you use pyenv you also avoid the issue of interfering with your system wide python install. We do recommend pyenv regardless if you use a virtual machine or are working directly on your operating system. After you have set up a proper python environment, make sure you have the newest version of pip installed with

```
1 $ pip install pip -U
```

To install Eve, you can say

```
2 $ pip install eve
```

As Eve also needs a backend database, and as MongoDB is an obvious choice for this, we will have to first install MongoDB. MongoDB is a Non-SQL database which helps to store light weight data

easily.

34.3.1 Ubuntu install of MongoDB

On Ubuntu you can install MongoDB as follows

```

3 $ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2930↔
   ↪ ADAE8CAF5059EE73BB4B58712A2291FA4AD5
4 $ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial↔
   ↪ /
5 mongodb-org/3.6 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org↔
   ↪ -3.6.list
6 $ sudo apt-get update
7 $ sudo apt-get install -y mongodb-org

```

34.3.2 OSX install of MongoDB

On OSX you can use the command

```

8 brew update.
9 brew install mongodb

```

34.3.3 Windows 10 Installation of MongoDB

Exercise 34.1 A student or student group of this class are invited to discuss on piazza on how to install mongoDB on Windows 10 and come up with an easy installation solution. Naturally we have the same 2 different ways on how to run mongo. In user space or in the system. As we want to make sure your computer stays secure. the solution must have an easy way on how to shut down the Mongo services.

An enhancement of this task would be to integrate this function into cloudmesh cmd5 with a command *mongo* that allows for easily starting and stopping the service from *cms*. ■

34.3.4 Database Location

After downloading Mongo, create the *db* directory. This is where the Mongo data files will live. You can create the directory in the default location and assure it has the right permissions. Make sure that the */data/db* directory has the right permissions by running

34.3.5 Verification

In order to check the MongoDB installation, please run the following commands in one terminal:

```

10 $ mkdir -p ~/cloudmesh/data/db
11 $ mongod --dbpath ~/cloudmesh/data/db

```

In another terminal we try to connect to mongo and issue a mongo command to show the databases:

```

12 $ mongo --host 127.0.0.1:27017
13 $ show databases

```

If they execute without errors, you have successfully installed MongoDB. In order to stop the running database instance run the following command. simply CTRL-C the running mongod process

34.3.6 Building a simple REST Service

In this section we will focus on creating a simple rest service. To organize our work we will create the following directory:

```
14 $ mkdir -p ~/cloudmesh/eve
15 $ cd ~/cloudmesh/eve
```

As Eve needs a configuration and it is read in by default from the file `settings.py` we place the following content in the file `~/cloudmesh/eve/settings.py`:

```
1 MONGO_HOST = 'localhost'
2 MONGO_PORT = 27017
3 MONGO_DBNAME = 'student_db'
4 DOMAIN = {
5     'student': {
6         'schema': {
7             'firstname': {
8                 'type': 'string'
9             },
10            'lastname': {
11                'type': 'string'
12            },
13            'university': {
14                'type': 'string'
15            },
16            'email': {
17                'type': 'string',
18                'unique': True
19            }
20            'username': {
21                'type': 'string',
22                'unique': True
23            }
24        }
25    }
26 }
27 RESOURCE_METHODS = ['GET', 'POST']
```

The DOMAIN object specifies the format of a student object that we are using as part of our REST service. In addition we can specify RESOURCE_METHODS which methods are activated for the REST service. This way the developer can restrict the available methods for a REST service. To pass along the specification for mongoDB, we simply specify the hostname, the port, as well as the database name.

Now that we have defined the settings for our example service, we need to start it with a simple python program. We could name that program anything we like, but often it is called simply `run.py`. This file is placed in the same directory where you placed the `settings.py`. In our case it is in the file `~/cloudmesh/eve/run.py` and contains the following python program:

```
1 from eve import Eve
2 app = Eve()
3
4 if __name__ == '__main__':
5     app.run()
```

This is the most minimal application for Eve, that uses the `settings.py` file for its configuration. Naturally, if we were to change the configuration file and for example change the `DOMAIN` and its schema, we would naturally have to remove the database previously created and start the service new. This is especially important as during the development phase we may frequently change the schema and the database. Thus it is convenient to develop necessary cleaning actions as part of a Makefile which we leave as easy exercise for the students.

Next, we need to start the services which can easily be achieved in a terminal while running the commands:

Previously we started the mongoDB service as follows:

```
16 $ mongod --dbpath ~/cloudmesh/data/db/
```

This is done in its own terminal, so we can observe the log messages easily. Next we start in another window the Eve service with

```
17 $ cd ~/cloudmesh/eve
18 $ python run.py
```

You can find the codes and commands up to this point in the following document.

- <https://github.com/cloudmesh/book/blob/dev/section/rest/rest-code.md>

34.3.7 Interacting with the REST service

Yet in another window, we can now interact with the REST service. We can use the commandline to save the data in the database using the REST api. The data can be retrieved in XML or in json format. Json is often more convenient for debugging as it is easier to read than XML.

Naturally, we need first to put some data into the server. Let us assume we add the user Albert Zweistein.

```
19 $ curl -H "Content-Type: application/json" -X POST \
20     -d '{"firstname":"Albert","lastname":"Zweistein", \
21         "school":"ISE","university":"Indiana University", \
22         "email":"albert@iu.edu", "username": "albert"}' \
23     http://127.0.0.1:5000/student/
```

To achieve this, we need to specify the header using **H** tag saying we need the data to be saved using json format. And **X** tag says the HTTP protocol and here we use POST method. And the tag **d** specifies the data and make sure you use json format to enter the data. Finally, the REST api endpoint to which we must save data. This allows us to save the data in a table called **student** in MongoDB within a database called **eve**.

In order to check if the entry was accepted in mongo and included in the server issue the following command sequence in another terminal:

```
1 $ mongo
```

Now you can query mongo directly with its shell interface

```
1 > show databases
2 > use student_db
3 > show tables # query the table names
4 > db.student.find().pretty() # pretty will show the json in a clear way
```

Naturally this is not really necessary for A REST service such as eve as we show you next how to gain access to the data via mongo while using REST calls. We can simply retrieve the information with the help of a simple URI:

```
24 $ curl http://127.0.0.1:5000/student?firstname=Albert
```

Naturally, you can formulate other URLs and query attributes that are passed along after the ?.

This will now allow you to develop sophisticated REST services. We encourage you to inspect the documentation provided by Eve to showcase additional features that you could be using as part of your efforts.

Let us explore how to properly use additional REST API calls. We assume you have MongoDB up and running. To query the service itself we can use the URI on the Eve port

```
25 $ curl -i http://127.0.0.1:5000
```

Your payload should look like the one below, if your output is not formatted like below try adding ?pretty=1

```
26 $ curl -i http://127.0.0.1:5000?pretty=1

1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 150
4 Server: Eve/0.7.6 Werkzeug/0.11.15 Python/2.7.5
5 Date: Wed, 17 Jan 2018 18:34:07 GMT
6
7 {
8   "_links": {
9     "child": [
10      {
11        "href": "student",
12        "title": "student"
13      }
14    ]
15  }
```

Remember that the API entry points include additional information such as links and a child, and href.

Exercise 34.2 Set up a python environment that works for your platform. Provide explicit reasons why anaconda and other prepackaged python versions have issues for cloud related activities. When may you use anaconda and when should you not use anaconda. Why would you want to use pyenv? ■

Exercise 34.3 What is the meaning and purpose of links, child, and href ■

Exercise 34.4 In this case how many child resources are available through our API? ■

Exercise 34.5 Develop a REST service with Eve and start and stop it ■

Exercise 34.6 Define curl calls to store data into the service and retrieve it. ■

Exercise 34.7 Write a Makefile and in it a target clean that cleans the data base. Develop additional targets such as start and stop, that start and stop the mongoDB but also the Eve REST service ■

Exercise 34.8 Issue the command

```
27 $ curl -i http://127.0.0.1:5000/people
```

What does the `_links` section describe?

What does the `_items` section describe?

```
1 {
2   "_items": [],
3   "_links": {
4     "self": {
5       "href": "people",
6       "title": "people"
7     },
8     "parent": {
9       "href": "/",
10      "title": "home"
11    }
12  },
13  "_meta": {
14    "max_results": 25,
15    "total": 0,
16    "page": 1
17  }
18 }
```

34.3.8 Creating REST API Endpoints

Next we want to enhance our example a bit. First, let us get back to the eve working directory with

```
1 $ cd ~/cloudmesh/eve
```

Add the following content to a file called **run2.py**

```
1 from eve import Eve
2 from flask import jsonify
3 import os
4 import getpass
5
6 app = Eve ()
7
8 @app.route('/student/albert')
9 def alberts_information():
10     data = {
11         'firstname': 'Albert',
12         'lastname': 'Zweistsein',
13         'university': 'Indiana University',
14         'email': 'albert@example.com'
15     }
16     try:
17         data['username'] = getpass.getuser()
18     except:
19         data['username'] = 'not-found'
20     return jsonify(**data)
21
22 if __name__ == '__main__':
23     app.run(debug=True, host="127.0.0.1")
```

After creating and saving the file. Run the following command to start the service

```
1 $ python run2.py
```

After running the command, you can interact with the service while entering the following url in the web browser:

```
1 http://127.0.0.1:5000/student/alberts
```

You can also open up a second terminal and type in it

```
1 curl http://127.0.0.1:5000/student/alberts
```

The following information will be returned:

```
1 {
2   "firstname": "Albert",
3   "lastname": "Zweistain",
4   "university": "Indiana University",
5   "email": "albert@example.com",
6   "username": "albert"
7 }
```

This example illustrates how easy it is to create REST services in python while combining information from a dict with information retrieved from the system. The important part is to understand the decorator **app.route**. The parameter specifies the route of the API endpoint which will be the

address appended to the base path, **http://127.0.0.1:5000**. It is important that we return a jsonified object, which can easily be done with the `jsonify` function provided by flask. As you can see the name of the decorated function can be anything you lok. The route specifies how we access it from the service.

34.3.9 REST API Output Formats and Request Processing

Another way of managing the data is to utilize class definitions and response types that we explicitly define.

If we want to create an object like Student, we can first define a python class. Create a file called **student.py**. Please, note the get method that returns simply the information in the dict for the class. It is not related to the REST get function.

```

1 class Student(object):
2     def __init__(self, firstname, lastname, university, email):
3         self.firstname = firstname
4         self.lastname = lastname
5         self.university = university
6         self.email = email
7         self.username = 'undefined'
8
9     def get(self):
10        return self.__dict__
11
12    def setUsername(self, name):
13        self.username = name
14        return name

```

Next we define a REST service with Eve as shown in the following listing

```

1 from eve import Eve
2 from student import Student
3 import platform
4 import psutil
5 import json
6 from flask import Response
7 import getpass
8
9 app = Eve()
10
11
12 @app.route('/student/albert', methods=['GET'])
13 def processor():
14     student = Student("Albert",
15                      "Zweistein",
16                      "Indiana University",
17                      "albert@example.edu")
18
19     response = Response()
20     response.headers["Content-Type"] = "application/json; charset=utf-8"
21
22     try:
23         student.setUsername(getpass.getuser())
24         response.headers["status"] = 200
25     except:

```

```

26     response.headers["status"] = 500
27
28     response.data = json.dumps(student.get())
29     return response
30
31 if __name__ == '__main__':
32     app.run(debug=True, host='127.0.0.1')

```

In contrast to our earlier example, we are not using the jsonify object, but create explicitly a response that we return to the clients. The response includes a header that we return the information in json format, a status of 200, which means the object was returned successfully, and the actual data.

34.3.10 WRONG: Consuming REST API Using a Client Application

TODO: This example is not tested. Please provide feedback and improve

In the Section 34.3.8 we created our own REST API application using Python Eve. Now once the service running, a we need to learn how to interact with it through clients.

First go back to the working folder:

```
1 $ cd ~/cloudmesh/eve
```

Here we create a new python file called **client.py**. The file include the following content.

```

1 import requests
2 import json
3
4 def get_all():
5     response = requests.get("http://127.0.0.1:5000/student")
6     print(json.dumps(response.json(), indent=4, sort_keys=True))
7
8
9 def save_record():
10    headers = {
11        'Content-Type': 'application/json'
12    }
13
14    data = '{"firstname":"Gregor",
15           "lastname":"von Laszewski",
16           "university": "Indiana University",
17           "email":"jane@iu.edu",
18           "username": "jane"}'
19
20    response = requests.post('http://localhost:5000/student/',
21                             headers=headers,
22                             data=data)
23    print(response.json())
24
25
26 if __name__ == '__main__':
27     save_record()
28     get_all()

```

Run the following command in a new terminal to execute the simple client by


```
1 $ python client.py
```

Here when you run this class for the first time, it will run successfully, but if you tried it for the second time, it will give you an error. Because we did set the email to be a unique field in the schema when we designed the settings.py file in the beginning. So if you want to save another record you must have entries with unique emails. In order to make this dynamic you can include a input reading by using the terminal to get the student data first and instead of the static data you can use the user input data from the terminal to get dynamic data. But for this exercise we don't expect that or any other form data functionality.

In order to get the saved data, you can comment the record saving function and uncomment the get all function. In python commenting is done by using #.

This client is using the **requests** python library to send GET, POST and other HTTP requests to the server so you can leverage build in methods to simplify your work.

The `get_all` function provides a way to get the output to the console with all the data in the student database. The `save_record` function provides a way to save data in the database. You can create dynamic functions in order to save dynamic data. However it may take some time for you to apply as exercise.

Exercise 34.9 Write a RESTful service to determine a useful piece of information off of your computer i.e. disk space, memory, RAM, etc. In this exercise what you need to do is use a python library to extract data about computer information mentioned above and send these information to the user once the user calls an API endpoint like `http://localhost:5000/performance/ram`, it must return the RAM value of the given machine. For each information like disk space, RAM, etc you can use an endpoint per each feature needed. As a tip for this exercise, use the `psutil` library in python to retrieve the data, and then get these information into a string then populate a class called `Computer` and try to save the object like wise. ■

34.3.11 HATEOAS

In the previous section we discussed the basic concepts about RESTful web service. Next we introduce you to the concept of HATEOAS

HATEOAS stands for Hypermedia as the Engine of Application State and this is enabled by the default configuration within Eve. It is useful to review the terminology and attributes used as part of this configuration. HATEOS explains how REST API endpoints are defined and it provides a clear description on how the API can be consumed through these terms:

_links . Links describe the relation of current resource being accessed to the rest of the resources.

It is like if we have a set of links to the set of objects or service endpoints that we are referring in the RESTful web service. Here an endpoint refers to a service call which is responsible for executing one of the CRUD operations on a particular object or set of objects. More on the links, the links object contains the list of serviceable API endpoints or list of services. When we are calling a GET request or any other request, we can use these service endpoints to execute different queries based on the user purpose. For instance, a service call can be used to insert data or retrieve data from a remote database using a REST API call. About databases we will discuss in detail in another chapter.

title . The title in the rest endpoint is the name or topic that we are trying to address. It describes the nature of the object by a single word. For instance student, bank-statement, salary, etc can be a title.

parent . The term parent refers to the very initial link or an API endpoint in a particular RESTful

web service. Generally this is denoted with the primary address like `http://example.com/api/v1/`. The term `href` refers to the url segment that we use to access the a particular REST API endpoint. For instance `“student?page=1”` will return the first page of student list by retrieving a particular number of items from a remote database or a remote data source. The full url will look like this, `“http://www.exampleapi.com/student?page=1”`.

In addition to these fields eve will automatically create the following information when resources are created as showcased ot

- <http://python-eve.org/features.html>

Field	Description
<code>_created</code>	item creation date.
<code>_updated</code>	item last updated on.
<code>_etag</code>	ETag, to be used for concurrency control and conditional requests.
<code>_id</code>	unique item key, also needed to access the individual item endpoint.

Pagination information can be included in the `_meta` field.

Filtering

CLients can submit query strings to the rest service to retrieve resources based on a filter. This also allows sorting of the results queried. One nice feature about using mongo as a backend database is that Eve not only allows python conditional expressions, but also mongo queries.

A number of examples to conduct such queries include:

```
1 curl -i -g http://eve-demo.herokuapp.com/people?where={%22lastname↵
   ↪ %22:%20%22Doe%22}
```

A python expression

```
1 curl -i http://eve-demo.herokuapp.com/people?where=lastname=="Doe"
```

34.3.12 Pretty Printing

Pretty printing is typically supported by adding the parameter `?pretty` or `?pretty=1`

If this does not work you can always use python to beautify a json output with

```
1 curl -i http://localhost/people?pretty
```

or

```
1 curl -i http://localhost/people | python -m json.tool
```

34.3.13 XML

If for some reason you like to retrieve the information in XML you can specify this for example through curl with an Accept header

```
1 curl -H "Accept: application/xml" -i http://localhost
```

34.3.14 Extensions to Eve

A number of extensions have been developed by the community. This includes `eve-swagger`, `eve-sqlalchemy`, `eve-elastic`, `eve-mongoengine`, `eve-neo4j`, `eve.net`, `eve-auth-jwt`, and `flask-sentinel`.

Naturally there are many more.

Students have the opportunity to pick one of the community extensions and provide a section for the handbook.

Exercise 34.10 Pick one of the extension, research it and provide a small section for the handbook so we add it. ■

34.4 Object Management with Eve and Evegenie

<http://python-eve.org/>

Eve makes the creation of a REST implementation in python easy. We will provide you with an implementation example that showcases that we can create REST services without writing a single line of code. The code for this is located at <https://github.com/cloudmesh/rest>

This code will have a master branch but will also have a dev branch in which we will add gradually more objects. Objects in the dev branch will include:

- virtual directories
- virtual clusters
- job sequences
- inventories

You may want to check our active development work in the dev branch. However for the purpose of this class the master branch will be sufficient.

34.4.1 Installation

First we have to install mongodb. The installation will depend on your operating system. For the use of the rest service it is not important to integrate mongodb into the system upon reboot, which is focus of many online documents. However, for us it is better if we can start and stop the services explicitly for now.

On ubuntu, you need to do the following steps:

TODO: TO BE CONTRIBUTED BY THE STUDENTS OF THE CLASS AS HOME-WORK

On windows 10, you need to do the following steps:

TODO: TO BE CONTRIBUTED BY THE STUDENTS OF THE CLASS AS HOME-WORK. If you elect Windows 10. You could be using the online documentation provided by starting it on Windows, or running it in a docker container.

On OSX you can use home-brew and install it with:

```
28 brew update
29 brew install mongodb
```

In future we may want to add ssl authentication in which case you may need to install it as follows:

```
30 brew install mongodb --with-openssl
```

34.4.2 Starting the service

We have provided a convenient Makefile that currently only works for OSX. It will be easy for you to adapt it to Linux. Certainly you can look at the targets in the makefile and replicate them one by one. Important targets are `deploy` and `test`.

When using the makefile you can start the services with:

```
31 make deploy
```

IT will start two terminals. IN one you will see the mongo service, in the other you will see the eve service. The eve service will take a file called `sample.settings.py` that is base on `sample.json` for the start of the eve service. The mongo servide is configured in such a way that it only accepts incoming connections from the local host which will be sufficient for our case. The mongo data is written into the `$(USER)/.cloudmesh` directory, so make sure it exists.

To test the services you can say:

```
32 make test
```

YOu will se a number of json text been written to the screen.

34.4.3 Creating your own objects

The example demonstrated how easy it is to create a mongodb and an eve rest service. Now let us use this example to create your own. For this we have modified a tool called `evegenie` to install it onto your system.

The original documentation for `evegenie` is located at:

- <http://evegenie.readthedocs.io/en/latest/>

However, we have improved `evegenie` while providing a commandline tool based on it. The improved code is located at:

- <https://github.com/cloudmesh/evegenie>

You clone it and install on your system as follows:

```
33 cd ~/github
34 git clone https://github.com/cloudmesh/evegenie
35 cd evegenie
36 python setup.py install
37 pip install .
```

This should install in your system `evegenie`. YOu can verify this by typing:

```
1 which evegenie
```

If you see the path `evegenie` is installed. With `evegenie` installed its usaage is simple:

```
1 $ evegenie
2
3 Usage:
4   evegenie --help
5   evegenie FILENAME
```

It takes a json file as input and writes out a settings file for the use in eve. Lets assume the file is called `sample.json`, than the settings file will be called `sample.settings.py`. Having the `evegenie`

programm will allow us to generate the settings files easily. You can include them into your project and leverage the Makefile targets to start the services in your project. In case you generate new objects, make sure you rerun eve, kill all previous windows in which you run eve and mongo and restart. In case of changes to objects that you have designed and run previously, you need to also delete the mongod database.

34.5 Towards cmd5 extensions to manage eve and mongo

Naturally it is of advantage to have in cms administration commands to manage mongo and eve from cmd instead of targets in the Makefile. Hence, we **propose** that the class develops such an extension. We will create in the repository the extension called admin and hope that students through collaborative work and pull requests complete such an admin command.

The proposed command is located at:

- <https://github.com/cloudmesh/rest/blob/master/cloudmesh/ext/command/admin.py>

It will be up to the class to implement such a command. Please coordinate with each other.

The implementation based on what we provided in the Make file seems straight forward. A great extension is to load the objects definitions or eve e.g. settings.py not from the class, but from a place in .cloudmesh. I propose to place the file at:

```
1 .cloudmesh/db/settings.py
```

the location of this file is used when the Service class is initialized with None. Prior to starting the service the file needs to be copied there. This could be achieved with a set command.

34.6 Responses

- <https://dzone.com/refcardz/rest-foundations-restful>

34.7 Django REST Framework

- <http://www.django-rest-framework.org/>

Django REST framework is a large toolkit to develop Web APIs. The developers of the framework provide the following reasons for using it:

“(1) The Web browsable API is a huge usability win for your developers. (2) Authentication policies including packages for OAuth1a and OAuth2. (3) Serialization that supports both ORM and non-ORM data sources. (4) Customizable all the way down - just use regular function-based views if you do not need the more powerful features. (5) Extensive documentation, and great community support. (6) Used and trusted by internationally recognised companies including Mozilla, Red Hat, Heroku, and Eventbrite.”

34.8 Rest Services with Swagger

Swagger <https://swagger.io/> is a tool for developing API specifications based on the OpenAPI Specification (OAS). It allows not only the specification, but the generation of code based on the specification in a variety of languages.

Swagger itself has a number of tools which together build a framework for developing REST services for a variety of languages.

34.9 Swagger Tools

The major Swagger tools of interest are:

Swagger Core includes libraries for working with Swagger specifications <https://github.com/swagger-api/swagger-core>.

Swagger Codegen allows to generate code from the specifications to develop Client SDKs, servers, and documentation. <https://github.com/swagger-api/swagger-codegen>

Swagger UI is an HTML5 based UI for exploring and interacting with the specified APIs <https://github.com/swagger-api/swagger-ui>

Swagger Editor is a Web-browser based editor for composing specifications using YAML <https://github.com/swagger-api/swagger-editor>

The developed APIs can be hosted and further developed on an online repository named Swagger-Hub <https://app.swaggerhub.com/home> The convenient online editor is available which also can be installed locally on a variety of operating systems including OSX, Linux, and Windows.

34.10 Swagger Community Tools

Exercise 34.11 notify us about other tools that you find and would like us to mention here. ■

34.10.1 Swagger Toolbox

Swagger toolbox is a utility that can convert json to swagger compatible yaml models. It is hosted online at

- <https://swagger-toolbox.firebaseio.com/>

The source code to this tool is available on github at

- <https://github.com/essuraj/swagger-toolbox>

It is important to make sure that the json model is properly configured. As such each datatype must be wrapped in “quotes” and the last element must not have a , behind it.

In case you have large models, we recommend that you gradually add more and more features so that it is easier to debug in case of an error. This tool is not designed to provide back a full featured OpenAPI, but help you getting started deriving one.

Let us look at a small example. Let us assume we want to create a REST service to execute a command on the remote service. We know this may not be a good idea if it is not properly secured,

so be extra careful. A good way to simulate this is to just use a return string instead of executing the command.

Let us assume the json schema looks like:

```
1 {
2   "host": "string",
3   "command": "string"
4 }
```


The output the swagger toolbox creates is

```
1 ---
2   required:
3     - "host"
4     - "command"
5   properties:
6     host:
7       type: "string"
8     command:
9       type: "string"
```

As you can see it is far from complete, but it could be used to get you started.

Exercise 34.12 Based on this tool develop a rest service to which you send a schema in JSON format from which you get back the YAML model. ■

34.11 REST Service Generation with Swagger

Swagger (36:02) 

In this subsection we are discussing how to use OpenAPI 2.0 and Swagger Codegen to define and develop a REST Service.

We assume you have been familiar with the concept of REST service, OpenAPI as discussed in section 34.1. In next section we will further look into the Swagger/OpenAPI 2.0 specification 34.12 and use a slight more complex example to walk you through the design of a RESTful service following the OpenAPI 2.0 specifications.

We will use a simple example to demonstrate the process of developing a REST service with Swagger/OpenAPI 2.0 specification and the tools related to it. The general steps are:

- Step 1 (Section 34.11.1). Define the REST service conforming to Swagger/OpenAPI 2.0 specification. It is a YAML document file with the basics of the REST service defined, e.g., what resources it has and what actions are supported.
- Step 2 (Section 34.11.2). Use Swagger Codegen to generate the server side stub code. Fill in the actual implementation of the business logic portion in the code.
- Step 3 (Section 34.11.3). Install the server side code and run it. The service will then be available.
- Step 4 (Section 34.11.4). Generate client side code. Develop code to call the REST service. Install and run to verify.

34.11.1 Step 1: Define Your REST Service

In this example we define a simple REST service that returns the hosting server's basic CPU info. The example specification in yaml is as follows:

```
1 swagger: "2.0"
2 info:
3   version: "0.0.1"
4   title: "cpuinfo"
5   description: "A simple service to get cpuinfo as an example of using ↔
6     ↔ swagger-2.0 specification and codegen"
7   termsOfService: "http://swagger.io/terms/"
8   contact:
9     name: "Cloudmesh REST Service Example"
10  license:
11    name: "Apache"
12 host: "localhost:8080"
13 basePath: "/api"
14 schemes:
15   - "http"
16 consumes:
17   - "application/json"
18 produces:
19   - "application/json"
20 paths:
21   /cpu:
22     get:
23       description: "Returns cpu information of the hosting server"
24       produces:
```



```

24     - "application/json"
25     responses:
26       "200":
27         description: "CPU info"
28         schema:
29           $ref: "#/definitions/CPU"
30 definitions:
31   CPU:
32     type: "object"
33     required:
34       - "model"
35     properties:
36       model:
37         type: "string"

```

34.11.2 Step 2: Server Side Stub Code Generation and Implementation

With the REST service having been defined, we can now generate the server side stub code easily.

Setup the Codegen Environment

You will need to [install the Swagger Codegen tool](#) if not yet done so. For OSX we recommend that you use the homebrew install via

```
1 brew install swagger-codegen
```

On Ubuntu you can install swagger as follows (update the version as needed):

```

1 mkdir ~/swagger
2 cd ~/swagger
3 wget https://oss.sonatype.org/content/repositories/releases/io/swagger/↔
↔ swagger-codegen-cli/2.3.1/swagger-codegen-cli-2.3.1.jar
4 alias swagger-codegen="java -jar ~/swagger/swagger-codegen-cli-2.3.1.jar"

```

Add the alias to your `.bashrc` or `.bash_profile` file. After you start a new terminal you can use in that terminal now the command

```
1 swagger-codegen
```

For other platforms you can just use the `.jar` file, which can be downloaded from [this link](#).

Also make sure Java 7 or 8 is installed in your system. To have a well defined location we recommend that you place the code in the directory `~/cloudmesh`. In this directory you will also place the `cpu.yaml` file.

Generate Server Stub Code

After you have the codegen tool ready, and with Java 7 or 8 installed in your system, you can run the following to generate the server side stub code:

```

1 swagger-codegen generate \
2   -i ~/cloudmesh/cpu.yaml \
3   -l python-flask \
4   -o ~/cloudmesh/swagger_example/server/cpu/flaskConnexion \
5   -D supportPython2=true

```

or if you have not created an alias

```

1 java -jar swagger-codegen-cli.jar generate \
2     -i ~/cloudmesh/cpu.yaml \
3     -l python-flask \
4     -o ~/cloudmesh/swagger_example/server/cpu/flaskConnexion \
5     -D supportPython2=true

```

In the specified directory under *flaskConnexion* you will find the generated python flask code, with python 2 compatibility. It is best to place the swagger code under the directory `~/cloudmesh` to have a location where you can easily find it. If you want to use python 3 make sure to chose the appropriate option. To switch between python 2 and python 3 we recommend that you use `pyenv` as discussed in our python section.

Fill in the actual implementation

Under the *flaskConnexion* directory, you will find a *swagger_server* directory, under which you will find directories with *models* defined and *controllers* code stub resides. The models code are generated from the definition in Step 1. On the controller code though, we will need to fill in the actual implementation. You may see a `default_controller.py` file under the *controllers* directory in which the resource and action is defined but yet to be implemented. In our example, you will find such a function definition which we list next:

```

1 def cpu_get(): # noqa: E501
2     """cpu_get
3
4     Returns cpu info of the hosting server # noqa: E501
5
6
7     :rtype: CPU
8     """
9     return 'do some magic!'

```

Please note the `do some magic!` string at the return of the function. This ought to be replaced with actual implementation what you would like your REST call to be really doing. In reality this could be some call to a backend database or datastore; a call to another API; or even calling another REST service from another location. In this example we simply retrieve the `cpu` model information from the hosting server through a simple python call to illustrate this principle. Thus you can define the following code:

```

1 import os, platform, subprocess, re
2
3 def get_processor_name():
4     if platform.system() == "Windows":
5         return platform.processor()
6     elif platform.system() == "Darwin":
7         command = "/usr/sbin/sysctl -n machdep.cpu.brand_string"
8         return subprocess.check_output(command, shell=True).strip()
9     elif platform.system() == "Linux":
10        command = "cat /proc/cpuinfo"
11        all_info = subprocess.check_output(command, shell=True).strip()
12        for line in all_info.split("\n"):
13            if "model name" in line:
14                return re.sub(".*model name.*:", "", line, 1)
15        return "cannot find cpuinfo"

```

And then change the `cpu_get()` function to the following:

```

1 def cpu_get(): # noqa: E501
2     """cpu_get
3
4     Returns cpu info of the hosting server # noqa: E501
5
6
7     :rtype: CPU
8     """
9     return CPU(get_processor_name())

```

Please note we are returning a CPU object as defined in the API and later generated by the codegen tool in the *models* directory.

It is best *not* to include the definition of `get_processor_name()` in the same file as you see the definition of `cpu_get()`. The reason for this is that in case you need to regenerate the code, your modified code will naturally be overwritten. Thus, to minimize the changes, we do recommend to maintain that portion in a different filename and import the function as to keep the modifications small.

At this step we have completed the server side code development.

34.11.3 Step 3: Install and Run the REST Service:

Now we can install and run the REST service. It is strongly recommended that you run this in a `pyenv` or a `virtualenv` environment.

Start a virtualenv:

In case you are not using `pyenv`, please use `virtual env` as follows:

```

1 virtualenv RESTServer
2 source RESTServer/bin/activate

```

Make sure you have the latest pip:

```

1 pip install -U pip

```

Install the requirements of the server side code:

```

1 cd ~/cloudmesh/swagger_example/server/cpu/flaskConnexion
2 pip install -r requirements.txt

```

Install the server side code package:

Under the same directory, run:

```

1 python setup.py install

```

Run the service

Under the same directory:

```

1 python -m swagger_server

```

You should see a message like this:

```

1 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)

```

Verify the service using a web browser:

Open a web browser and visit:

- <http://localhost:8080/api/cpu>

to see if it returns a json object with cpu model info in it.

Assignment: How would you verify that your service works with a curl call?

34.11.4 Step 4: Generate Client Side Code and Verify

In addition to the server side code swagger can also create a client side code.

Client side code generation:

Generate the client side code in a similar fashion as we did for the server side code:

```
1 java -jar swagger-codegen-cli.jar generate \
2     -i ~/cloudmesh/cpu.yaml \
3     -l python \
4     -o ~/cloudmesh/swagger_example/client/cpu \
5     -D supportPython2=true
```

Install the client side code package:

Although we could have installed the client in the same python pyenv or virtualenv, we showcase here that it can be installed in a completely different environment. That would make it even possible to use a python 3 based client and a python 2 based server showcasing interoperability between python versions (although we just use python 2 here). Thus we create a new python virtual environment and conduct our install.

```
1 virtualenv RESTClient
2 source RESTClient/bin/activate
3 pip install -U pip
4 cd swagger_example/client/cpu
5 pip install -r requirements.txt
6 python setup.py install
```

Using the client API to interact with the REST service

Under the directory *swagger_example/client/cpu* you will find a README.md file which serves as an API documentation with example client code in it. E.g., if we save the following code into a .py file:

```
1 from __future__ import print_function
2 import time
3 import swagger_client
4 from swagger_client.rest import ApiException
5 from pprint import pprint
6 # create an instance of the API class
7 api_instance = swagger_client.DefaultApi()
8
9 try:
10     api_response = api_instance.cpu_get()
11     pprint(api_response)
12 except ApiException as e:
```

```
13 print("Exception when calling DefaultApi->cpu_get: %s\n" % e)
```

We can then run this code to verify the calling to the REST service actually works. We are expecting to see a return similar to this:

```
1 {'model': 'Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz'}
```

Obviously, we could have applied additional cleanup of the information returned by the python code, such as removing duplicated spaces.

34.11.5 Towards a Distributed Client Server

Although we develop and run the example on one localhost machine, you can separate the process into two separate machines. E.g., on a server with external IP or even DNS name to deploy the server side code, and on a local laptop or workstation to deploy the client side code. In this case please make changes on the API definition accordingly, e.g., the **host** value.

34.11.6 Exercises

Exercise 34.13 In Section 34.11.1, we introduced a schema. The question relates to termsOfService: Investigate what the termOfService attribute is and suggest a better value. Discuss on piazza. ■

Exercise 34.14 In Section 34.11.1, we introduced a schema. The question relates to model: What is the meaning of model under the definitions ■

Exercise 34.15 In Section 34.11.1, we introduced a schema. The question relates to \$ref: what is the meaning of the \$ref. Discuss on piazza, come up with a student answer in class. ■

Exercise 34.16 In Section 34.11.1, we introduced a schema. What does the response 200 mean. Do you need other responses? ■

Exercise 34.17 After you have gone through the entire section and verified it works for you add create a more sophisticated schema and add more attributes exposing more information from your system. ■

Exercise 34.18 How can you for example develop a rest service that exposes portions of your file system serving large files, e.g. their filenames and their size? How would you download these files? Would you use a rest service, or would you register an alternative service such as ftp, DAV, or others? Please discuss in piazza. Note this will be a helping you to prepare a larger assignment. Think about this first before you implement. ■

Exercise 34.19 You can try expand the API definition with more resources and actions included. E.g., to include more detailed attributes in the CPU object and to have those information provided in the actual implementation as well. Or you could try defining totally different resources. ■

Exercise 34.20 The codegen tool provides a convenient way to have the code stubs ready, which frees the developers to focus more on the API definition and the real implementation of the business logics. Try with complex implementation on the back end server side code to interact with a database/datastore or a 3rd party REST service. ■

Exercise 34.21 For advanced python users, you can naturally use function assignments to replace the `cpu_get()` entirely even after loading the instantiation of the server. However, this is not needed. If you are an advanced python developer, please feel free to experiment and let us know how you suggest to integrate things easily. ■

34.12 Swagger Specification

Swagger provides through its specification the definition of REST services through a YAML or JSON document.

When following the API-specification-first approach to define and develop a RESTful service, the first and foremost step is to define the API conforming to the OpenAPI specification, and then using codegen tools to conveniently generate server side stub code, client code, documentations, in the language you desire. In Section 34.11 we have introduced the codegen tool and how to use that to generate server side and client side code and documentation. In this Section 34.12.1 we will use a slightly more complex example to show how to define an API following the OpenAPI 2.0 specification. The example is to retrieve virtual cluster (VC) object from the server.

The OpenAPI Specification is formerly known as Swagger RESTful API Documentation Specification. It defines a specification to describe and document a RESTful service API. It is also known under version 3.0 of swagger. However, as the tools for 3.0 are not yet completed, we will continue for now to use version swagger 2.0, till the transition has been completed. This is especially of importance, as we need to use the swagger codegen tool, which currently support only up to specification v2. Hence we are at this time using OpenAPI/Swagger v2.0 in our example. There are some structure and syntax changes in v3, while the essence is very similar. For more details of the changes between v3 and v2, please refer to A document published on the Web titled [Difference between OpenAPI 3.0 and Swagger 2.0](#).

You can write the API definition in json for yaml format. Let us discuss this format briefly and focus on yaml as it is easier to read and maintain.

On the root level of the yaml document we see fields like *swagger*, *info*, and so on. Among these fields, *swagger*, *info*, and *path* are **required**. Their meaning is as follows:

swagger specifies the version number. IN our case a string value '2.0' is used as we are writing the definition conforming to the v2.0 specification.

info defines metadata information related to the API. E.g., the API *version*, *title* and *description*, *termsOfService* if applicable, *contact* information and *license*, etc. Among these attributes, *version* and *title* are required while others are optional.

path defines the actual endpoints of the exposed RESTful API service. Each endpoint has a *field pattern* as the key, and a *Path Item Object* as the value. In this example we have defined */vc* and */vc/{id}* as the two service endpoints. They will be part of the final service URL, appended after the service *host* and *basePath*, which will be explained later.

Let us focus on the *Path Item Object*. It contains one or more supported *operations* on the service endpoint. An *operation* is keyed by a valid HTTP operation verb, e.g., one of **get**, **put**, **post**, **delete**, or **patch**. It has a value of *Operation Object* that describes the operations in more detail.

The *Operation Object* will always **require** a *Response Object*. A *Response Object* has a *HTTP status code* as the key, e.g., **200** as successful return; **40X** as authentication and authorization related errors; and **50x** as other server side servers. It can also has a default response keyed by **default** for undeclared http status return code. The *Response Object* value has a **required** *description* field, and if anything is returned, a *schema* indicating the object type to be returned, which could be a primitive type, e.g., *string*, or an *array* or customized *object*. In case of *object* or an *array of object*, use *\$ref* to point to the definition of the object. In this example, we have

```
1 $ref: "#/definitions/VC"
```

to point to the *VC* definition in the *definitions* section in the same specification file, which will be explained later.

Besides the required field, the *Operation Object* can have *summary* and *description* to indicate what the operation is about; and *operationId* to uniquely identify the operation; and *consumes* and *produces* to indicate what MIME types it expects as input and for returns, e.g., *application/json* in most modern RESTful APIs. It can further specify what input parameter is expected using *parameters*, which requires a *name* and *in* fields. *name* specifies the name of the parameter, and *in* specifies from where to get the parameter, and its possible values are *query*, *header*, *path*, *formData* or *body*. In this example in the */vc/{id}* path we obtain the *id* parameter from the URL path wo it has the *path* value. When the *in* has *path* as its value, the *required* field is required and has to be set as *true*; when the *in* has value other than *body*, a *type* field is required to specify the type of the parameter.

While the three root level fields mentioned above are required, in most cases we will also use other optional fields.

host to indicate where the service is to be deployed, which could be *localhost* or a valid IP address or a DNS name of the host where the service is to be deployed. If other port number other than *80* is to be used, write the port number as well, e.g., *localhost:8080*.

schemas to specify the transfer protocol, e.g, *http* or *https*.

basePath to specify the common base URL to be append after the *host* to form the base path for all the endpoints, e.g., */api* or */api/1.0/*. In this example with the values specified we would have the final service endpoints *http://localhost:8080/api/vcs* and *http://localhost:8080/api/vc/{id}* by combining the *schemas*, *host*, *basePath* and *paths* values.

consumes and produces can also be specified on the top level to specify the default MIME types of the input and return if most *paths* and the defined operations have the same.

definitions as used in in the *paths* field, in order to point to a customized object definition with a *\$ref* keyword.

The *definitions* field really contains the object definition of the customized objects involved in the API, similar to a class definition in any Object Oriented programming language. In this example, we defined a *VC* object, and hierarchically a *Node* object. Each object defined is a type of *Schema Object* in which many field could be used to specify the object (see details in the REF link at top of the document), but the most frequently used ones are:

type to specify the type, and in the customized definition case the value is mostly *object*.

required field to list the names of the required attributes of the object.

properties has the detailed information of each attribute/property of the object, e.g, *type*, *format*.

It also supports hierarchical object definition so a property of one object could be another customized object defined elsewhere while using *schema* and *\$ref* keyword to point to the definition. In this example we have defined a *VC*, or virtual cluster, object, while it contains another object definition of

Node as part of a cluster.

34.12.1 The Virtual Cluster example API Definition

Terminology

VC A virtual cluster, which has one Front-End (FE) management node and multiple compute nodes. A VC object also has *id* and *name* to identify the VC, and *nnodes* to indicate how many compute nodes it has.

FE A management node from which to access the compute nodes. The FE node usually connects

to all the compute nodes via private network.

Node A computer node object that the info *ncores* to indicate number of cores it has, and *ram* and *localdisk* to show the size of RAM and local disk storage.

Specification

```

1 swagger: "2.0"
2 info:
3   version: "1.0.0"
4   title: "A simple Virtual Cluster"
5   description: "A simple service for a Virtual CLuster as a test of using ↵
6     ↵ swagger-2.0 specification and codegen"
7   termsOfService: "http://swagger.io/terms/"
8   contact:
9     name: "IU ISE software and system team"
10  license:
11    name: "Apache"
12  host: "localhost:8080"
13  basePath: "/api"
14  schemes:
15    - "http"
16  consumes:
17    - "application/json"
18  produces:
19    - "application/json"
20  paths:
21    /vcs:
22      get:
23        description: "Returns all VCs from the system that the user has ↵
24          ↵ access to"
25        produces:
26          - "application/json"
27        responses:
28          "200":
29            description: "A list of VCs."
30            schema:
31              type: "array"
32              items:
33                $ref: "#/definitions/VC"
34    /vcs/{id}:
35      get:
36        description: "Returns all VCs from the system that the user has ↵
37          ↵ access to"
38        operationId: getVCById
39        parameters:
40          - name: id
41            in: path
42            description: ID of VC to fetch
43            required: true
44            type: string
45        produces:
46          - "application/json"
47        responses:
48          "200":
49            description: "The vc with the given id."
50            schema:

```

```
48         $ref: "#/definitions/VC"
49     default:
50         description: unexpected error
51         schema:
52             $ref: '#/definitions/Error'
53 definitions:
54     VC:
55         type: "object"
56         required:
57             - "id"
58             - "name"
59             - "nnodes"
60             - "FE"
61             - "computes"
62         properties:
63             id:
64                 type: "string"
65             name:
66                 type: "string"
67             nnodes:
68                 type: "integer"
69                 format: "int64"
70             FE:
71                 type: "object"
72                 schema:
73                     $ref: "#/definitions/Node"
74             computes:
75                 type: "array"
76                 items:
77                     $ref: "#/definitions/Node"
78             tag:
79                 type: "string"
80     Node:
81         type: "object"
82         required:
83             - "ncores"
84             - "ram"
85             - "localdisk"
86         properties:
87             ncores:
88                 type: "integer"
89                 format: "int64"
90             ram:
91                 type: "integer"
92                 format: "int64"
93             localdisk:
94                 type: "integer"
95                 format: "int64"
96     Error:
97         required:
98             - code
99             - message
100        properties:
101            code:
102                type: integer
103                format: int32
```

```
104     message:  
105         type: string
```

34.12.2 Refernces

[The official OpenAPI 2.0 Documentation](#)



35. MQTT

With the increase importance of cloud computing and the increased number of edge devices and their applications, such as sensor networks, it is crucial to enable fast communication between the sensing devices and actuators, which may not be directly connected, as well as cloud services that analyze the data. To allow services that are built on different software and hardware platforms to communicate, a data agnostic, fast and service is needed. In addition to communication, the data generated by these devices, services, and sensors must be analyzed. Security aspects to relay this data is highly important. We will introduce a service called MQTT, which is a common, easy to use, queuing protocol that helps meet these requirements.

35.1 Introduction

As Cloud Computing and Internet of Things (IoT) applications and sensor networks become commonplace and more and more devices are being connected, there is an increased need to allow these devices to communicate quickly and securely. In many cases these edge devices have very limited memory and need to conserve power. The computing power on some of these devices is so limited that the sensory data need to be analyzed remotely. Furthermore, they may not even have enough computing capacity to process traditional HTTP web requests efficiently [115][295] or these traditional Web-based services are too resource hungry. Monitoring the state of a remotely located sensor using HTTP would require sending requests and receiving responses to and from the device frequently, which may not be efficient on small circuits or embedded chips on edge computing sensors [115].

Message Queue Telemetry Transport (MQTT) is a lightweight machine to machine (M2M) messaging protocol, based on a client/server publish-subscribe model. It provides a simple service allowing us to communicate between sensors, and services based on a subscription model.

MQTT was first developed in 1999 to connect oil pipelines [295]. The protocol has been designed

to be used on top of TCP/IP protocol in situations where network bandwidth, and available memory are limited allowing low power usage. However, as it is implemented on top of TCP/IP it is reliable in contrast to other protocols such as UDP. It allows efficient transmission of data to various devices listening for the same event, and is scalable as the number of devices increase [734][430].

The current support for MQTT is conducted as part of the Eclipse Paho project [180]. As MQTT is a protocol many different clients in various languages exist. This includes languages such as Python, C, Java, Lua, and many more.

The current standard of MQTT is available at

<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf>

35.2 Publish Subscribe Model

MQTT works via a publish-subscribe model that contains 3 entities: (1) a publisher, that sends a message, (2) a broker, that maintains queue of all messages based on topics and (3) multiple subscribers that subscribe to various topics they are interested in [657].

This allows for decoupling of functionality at various levels. The publisher and subscriber do not need to be close to each other and do not need to know each others identity. They need only to know the broker, as the publisher and the subscribers do not have to be running either at the same time nor on the same hardware [427].

Ready to use implementation exist to be deployed as brokers in the users application frameworks. A broker is a service that relays information between the client and servers. Common brokers include the open source Mosquito broker [430] and the Eclipse Paho MQTT Broker [180].

35.2.1 Topics

MQTT implements a hierarchy of topics that are relates to all messages. These topics are recognised by strings separated by a forward-slash (/), where each part represents a different topic level. This is a common model introduced in file systems but also in internet URLs.

A topic looks therefore as follows: *topic-level0/topic-level1/topic-level2*.

Subscribers can subscribe to different topics via the broker. Subscribing to *topic-level0* allows the subscriber to receive all messages that are associated with topics that start with *topic-level0*. This allows subscribers to filter what messages to receive based on the topic hierarchy. Thus, when a publisher publishes a message related to a topic to the broker, the message is forwarded to all the clients that have subscribed to the topic of the message or a topic that has a lower depth of hierarchy [427] [657].

This is different from traditional point-to-point message queues as the message is forwarded to multiple subscribers, and allows for flexibility of dealing with subscribed topics not only on the server but also on the subscriber side [427]. The basic steps in a MQTT client subscriber application include to (1) connect to the broker, (2) subscribe to some topics, (3) wait for messages and (4) perform the appropriate action when a certain message is received [734].

35.2.2 Callbacks

One of the advantages of using MQTT is that it supports asynchronous behaviour with the help of callbacks. Both the publisher and subscriber can use non-blocking callbacks to act upon message

exchanges. [427][484].

For example, the `paho-mqtt` package for python provides callbacks methods including `on-connect()`, `on-message()` and `on-disconnect()`, which are invoked when the connection to the broker is complete, a message is received from the broker, and when the client is disconnected from the broker respectively. These methods are used in conjunction with the `loop-start()` and `loop-end()` methods which start and end an asynchronous loop that listens for these events invoking the relevant callbacks. Hence it frees the services to perform other tasks [484] when no messages are available, thus reducing overhead.

35.2.3 Quality of Service

MQTT has been designed to be flexible allowing for the change of quality of service (QoS) as desired by the application. Three basic levels of QoS are supported by the protocol: Atmost-once (QoS level 0), Atleast-once (QoS level 1) and Atmost-once (QoS level 2) [425, 484].

QoS level 0: The QoS level of 0 is used in applications where some dropped messages may not affect the application. Under this QoS level, the broker forwards a message to the subscribers only once and does not wait for any acknowledgement [425][484].

QoS Level 1: The QoS level of 1 is used in situations where the delivery of all messages is important and the subscriber can handle duplicate messages. Here the broker keeps on resending the message to a subscriber after a certain timeout until the first acknowledgement is received.

QoS Level 3: A QoS level of 3 is used in cases where all messages must be delivered and no duplicate messages should be allowed. In this case the broker sets up a handshake with the subscriber to check for its availability before sending the message [425, 484].

The various levels of quality of service allow the use of this protocol with different service level expectations.

35.3 Secure MQTT Services

MQTT specification uses TCP/IP to deliver the messaged to the subscribers, but it does not provide security by default to enable resource constrained IoT devices. “It allows the use of username and password for authentication, but by default this information is sent as plain text over the network, making it susceptible to man-in-the middle attacks” [426, 481]. Therefore, to support sensitive applications additional security measures need to be integrated through other means. This may include for example the use of Virtual Private Networks (VPNs), Transport Layer Security, or application layer security [426].

35.3.1 Using TLS/SSL

Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are cryptographic protocols that establish a the identity of the server and client with the help of a handshake mechanism which uses trust certificates to establish identities before encrypted communication can take place [151]. If the handshake is not completed for some reason, the connection is not established and no messages are exchanged [426]. “Most MQTT brokers provide an option to use TLS instead of plain TCP and port 8883 has been standardized for secured MQTT connections” [481].

Using TLS/SSL security however comes at an additional cost. If the connections are short-lived then most of the time is spent in verifying the security of the handshake itself, which in addition

to using time for encryption and decryption, may take up few kilobytes of bandwidth. In case the connections are short-lived, temporary session IDs and session tickets can be used as alternative to resume a session instead of repeating the handshake process. If the connections are long term, the overhead of the handshake is negligible and TLS/SSL security should be used [426, 481].

35.3.2 Using OAuth

OAuth is an open protocol that allows access to a resource without providing unencrypted credentials to the third party. Although MQTT protocol itself does not include authorization, many MQTT brokers include authorization as an additional feature [151]. OAuth2.0 uses JSON Web Tokens which contain information about the token and the user and are signed by a trusted authorization server [428].

When connecting to the broker this token can be used to check whether the client is authorised to connect at this time or not. Additionally the same validations can be used when publishing or subscribing to the broker. The broker may use a third party resource such as LDAP (lightweight directory access protocol) to look up authorizations for the client [428]. Since there can be a large number of clients and it can become impractical to authorize everyone, clients may be grouped and the authorizations may be checked for each group [151].

35.4 Integration with Other Services

As the individual IoT devices perform their respective functions in the sensor network, a lot of data is generated which needs to be processed. MQTT allows easy integration with other services, that have been designed to process this data. Let us provide some examples of MQTT integration into other Services.

Apache Storm. Apache storm is a distributed processing system that allows real time processing of continuous data streams, much like Hadoop works for batch processing [37]. Apache storm can be easily integrated with MQTT as shown in [605] to get real time data streams and allow analytics and online machine learning in a fault tolerant manner [738].

ELK stack. ELK stack (elastic-search, logstash and kibana) is an opensource project designed for scalability which contains three main software packages, the *elastic-search* search and analytics engine, *logstash* which is a data collection pipeline and *kibana* which is a visualization dashboard [189]. Data from an IoT network can be collected, analysed and visualized easily with the help of the ELK stack as shown in [202] and [201].

35.5 MQTT in Production

When using optimized MQTT broker services, MQTT can be utilized for enterprise and production environments. A good example is the use of EMQ (Erlang MQTT Broker) that provides a highly scalable, distributed and reliable MQTT broker for enterprise-grade applications [194].

35.6 Simple Usecase

In this example we are demonstrating how to set up a MQTT broker, a client and a subscriber.

- <https://github.com/bigdata-i523/sample-hid000/tree/master/experiment/mqtt>

A test program that starts a MQTT broker and client is provided next, showcasing how simple the interactions are while using a higher level API such as provided through the python client library of Paho.

```

1 import paho.mqtt.client as mqtt
2 import time
3
4
5 def on_message(client, userdata, message):
6     print("message received ", str(message.payload.decode("utf-8")))
7     print("message topic=", message.topic)
8     print("message qos=", message.qos)
9     print("message retain flag=", message.retain)
10
11 def on_log(client, userdata, level, buf):
12     print("log: ",buf)
13
14 broker_address="localhost"
15 # broker_address="test.mosquitto.org"
16 # broker_address="broker.hivemq.com"
17 # broker_address="iot.eclipse.org"
18
19 print("creating new instance")
20 client = mqtt.Client("i523") #create new instance
21 client.on_log=on_log
22 client.on_message=on_message #attach function to callback
23
24 print("connecting to broker")
25 client.connect(broker_address) #connect to broker
26 client.loop_start() #start the loop
27
28 print("Subscribing to topic","robot/leds/led1")
29 client.subscribe("robot/leds/led1")
30
31 print("Publishing message to topic","robot/leds/led1")
32 client.publish("robot/leds/led1","OFF")
33
34 time.sleep(4) # wait
35 client.loop_stop() #stop the loop

```

35.7 IoT Use Case

MQTT can be used in a variety of applications. This section explores a particular use case of the protocol. A small network was set up with three devices to simulate an IoT environment, and actuators were controlled with the help of messages communicated over MQTT.

The code for the project is available at

- <https://github.com/bigdata-i523/hid201/tree/master/experiment/mqtt>

35.7.1 Requirements and Setup

The setup used three different machines. A laptop or a desktop running the MQTT broker, and two raspberry pis configured with raspbean operating system. Eclipse Paho MQTT client was setup on

each of the raspberry pis [484]. Additionally all three devices were connected to an isolated local network.

GrovePi shields for the raspberry pis, designed by Dexter Industries were used on each of the raspberry pis to connect the actuators as they allow easy connections of the raspberry pi board [324]. The actuators used were Grove relays [606] and Grove LEDs [607] which respond to the messages received via MQTT.

To control the leds and relays, the python library `cloudmesh.pi` [137], developed at Indiana University was used. The library consists of interfaces for various IoT sensors and actuators and can be easily used with the grove modules.

35.7.2 Results

The two raspberry pis subscribe connect to the broker and subscribe with different topics. The raspberry pis wait for any messages from the broker. A publisher program that connects to the broker publishes messages to the broker for the topics that the two raspberry pis had registered. Each raspberry pi receives the corresponding message and turns the LEDs or relays on or off as per the message.

On a local network this process happens in near real time and no delays were observed. Eclipse IoT MQTT broker (*iot.eclipse.org*) was also tried which also did not result in any significant delays.

Thus it is observed that two raspberry pis can be easily controlled using MQTT. This system can be extended to include arbitrary number of raspberry pis and other devices that subscribe to the broker. If a device fails, or the connection from one device is broken, other devices are not affected and continue to perform the same.

This project can be extended to include various other kinds of sensors and actuators. The actuators may subscribe to topics to which various sensors publish their data and respond accordingly. The data of these sensors can be captured with the help of a data collector which may itself be a different subscriber, that performs analytics or visualizations on this data.

35.8 Conclusion

We see that as the number of connected devices increases and their applications become commonplace, MQTT allows different devices to communicate with each other in a data agnostic manner. MQTT uses a publish-subscribe model and allows various levels of quality of service requirements to be fulfilled. Although MQTT does not provide data security by default, most brokers allow the use of TLS/SSL to encrypt the data. Additional features may be provided by the broker to include authorization services. MQTT can be easily integrated with other services to allow collection and analysis of data. A small environment was simulated that used MQTT broker and clients running on raspberry pis to control actuators

35.9 Exercises

Exercise 35.1 Develop a temperature broker, that collects the temperature from a number of machines and clients can subscribe to the data and visualize it. ■

Exercise 35.2 Develop a CPU load broker, that collects the cpu load from a number of machines and clients can subscribe to the data and visualize it. ■

Exercise 35.3 Develop a broker with a variety of topics that collects data from a Raspberry Pi or Raspberry PI cluster and visualize it. ■

Exercise 35.4 Explore hosted services for MQTT while at the same time remembering that they could pose a security risk. Can any of the online services be used to monitor a cluster safely? ■



36. GraphQL

TODO: To be contributed by Averill Cate hid-sp18-???



Cloud Container Technologies

37	Introduction to Containers	451
37.1	Motivation - Microservices	
37.2	Motivation - Serverless Computing	
37.3	Docker	
37.4	Docker and Kubernetes	
37.5	QEMU and KVM	
37.6	Resources	
37.7	Introduction to Docker	
37.8	Docker Survey	
38	Running Docker Locally	455
38.1	Installation for OSX	
38.2	Installation for Ubuntu	
38.3	Draft: Installation for Windows 10	
38.4	Testing the Install	
39	Docker and Docker Swarm on FutureSystems	459
39.1	Getting Access	
39.2	Creating a service and deploy to the swarm cluster	
39.3	Dockerized REST Service	
40	Introduction to Kubernetes	469
40.1	Topics Covered and Learning Outcome	
40.2	What is Kubernetes?	
40.3	What are containers?	
40.4	Terminology	
40.5	Kubernetes Architecture	
40.6	Minikube	
40.7	Interactive Tutorial Online	
40.8	Using Kubernetes on FutureSystems	
41	Apache Hadoop using Docker	477
41.1	Draft: Creating the Hadoop Container	
41.2	Hadoop from Docker	
41.3	Start a Hadoop container	
41.4	Statistical Example with Hadoop	
41.5	Conclusion	
42	Apache Hadoop Latest (3.0.1) using Docker	485
42.1	Draft: Building Hadoop 3.0.1 using Docker	
42.2	Start a Hadoop Container with Interactive Shell	
42.3	Examples	
42.4	Draft: Apache Spark with Docker	
42.5	Draft: Multinode Hadoop Cluster Deployment with Docker Swarm	
43	Exercises	495
43.1	Docker Swarm Tutorial	
43.2	Docker Swarm on Google Compute Engine	
43.3	Single Node Hadoop	
43.4	Single Node Hadoop	
43.5	Spark Cluster	
44	Docker Ecosystem	497
44.1	Docker Hub	



37. Introduction to Containers

This section covers an introduction to containers that is split up into four parts. We discuss microservices, serverless computing, Docker, and kubernetes.


37.1 Motivation - Microservices

We discuss the motivation for containers and contrast them to virtual machines. Additionally we provide a motivation for containers as they can be used to microservices.

Container A (11:01) 

37.2 Motivation - Serverless Computing

We enhance our motivation while contrasting containers and microservices while relating them to serverless computing. We anticipate that serverless computing will increase in importance over the next years

Container B (15:08) 

37.3 Docker


In order for us to use containers, we go beyond the historical motivation that was introduced in a previous section and focus on Docker a predominant technology for containers on Windows, Linux, and OSX

Container C (40:09) 

[section/container/container.tex](#)

37.4 Docker and Kubernetes

We continue our discussion about docker and introduce kubernetes, allowing us to run multiple containers on multiple servers building a cluster of containers.

Container D (50:14) 

37.5 QEMU and KVM

We discuss the underlying virtualization technologies used by OpenStack and public cloud providers, such as AWS, Azure, and Google Cloud.

QEMU KVM (50) 

37.6 Resources

37.6.1 Container Orchestration Tools: Compare Kubernetes vs Docker Swarm

- <https://platform9.com/blog/compare-kubernetes-vs-docker-swarm/>

37.6.2 Gentle introduction to Containers

- <https://www.slideshare.net/jpetazzo/introduction-docker-linux-containers-lxc>

37.6.3 Tutorialspoint

- <https://www.tutorialspoint.com/docker/index.htm>
- https://www.tutorialspoint.com/docker/docker_tutorial.pdf
- https://www.tutorialspoint.com/docker/docker_pdf_version.htm

37.7 Intorduction to Docker

Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud. Today's businesses are under pressure to digitally transform but are constrained by existing applications and infrastructure while rationalizing an increasingly diverse portfolio of clouds, datacenters and application architectures. Docker enables true independence between applications and infrastructure and developers and IT ops to unlock their potential and creates a model for better collaboration and innovation. An overview of docker is provided at

- <https://docs.docker.com/engine/docker-overview/>

Image Source <https://www.docker.com/sites/default/files/Package%20software%40x2.png>

37.8 Docker Survey

In 2016 Docker Inc. surveyed over 500 Docker developers and operations experts in various phases of deploying container-based technologies. The result is available in the *The Docker Survey 2016*.

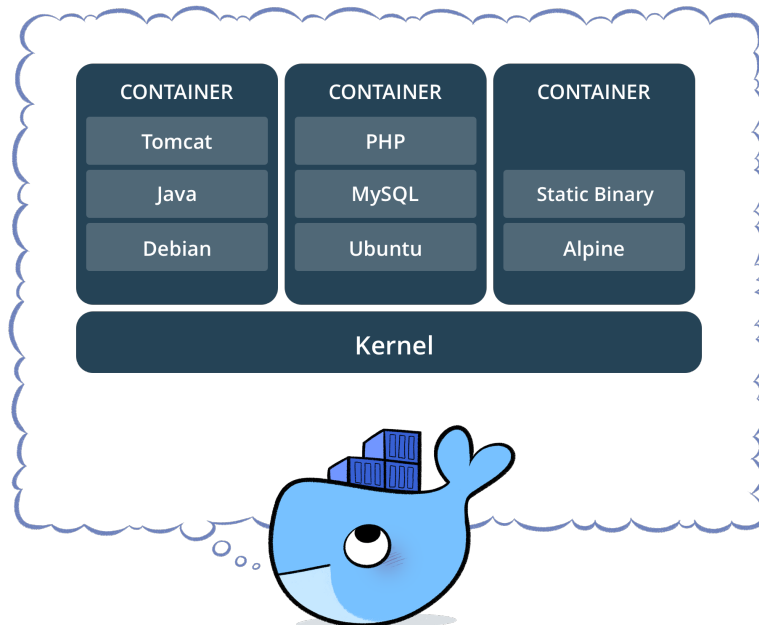


Figure 37.1: Docker Containers

- <https://www.docker.com/survey-2016>

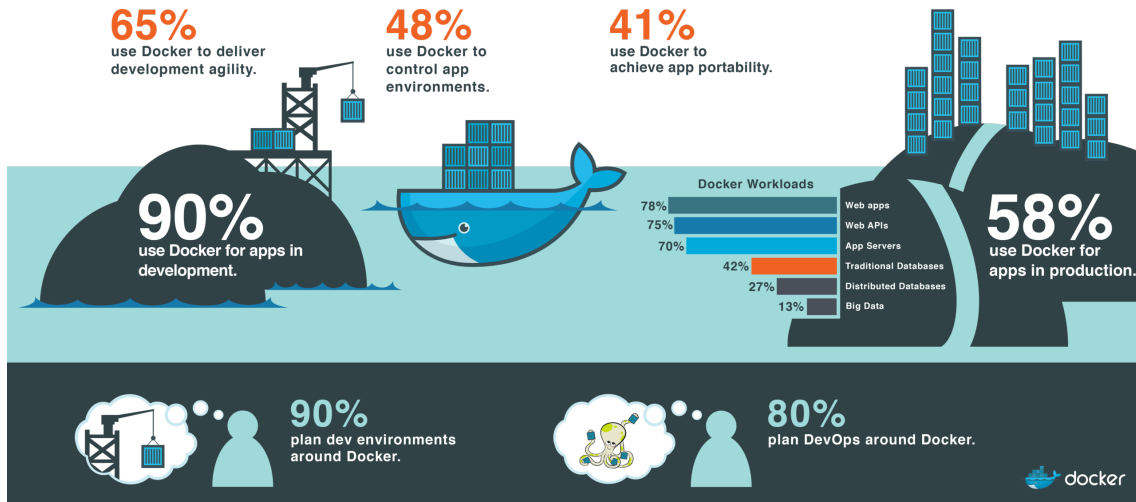


Figure 37.2: Docker Survey Results 2016

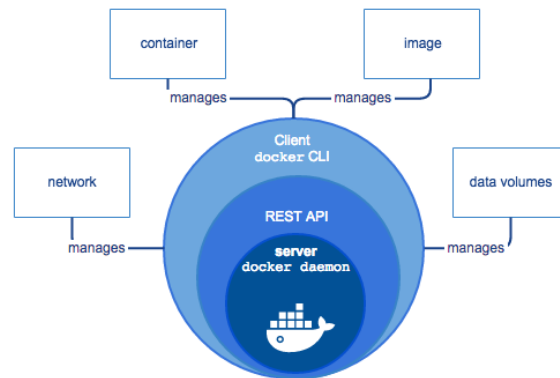


Figure 37.3: Docker Engine Component Flow

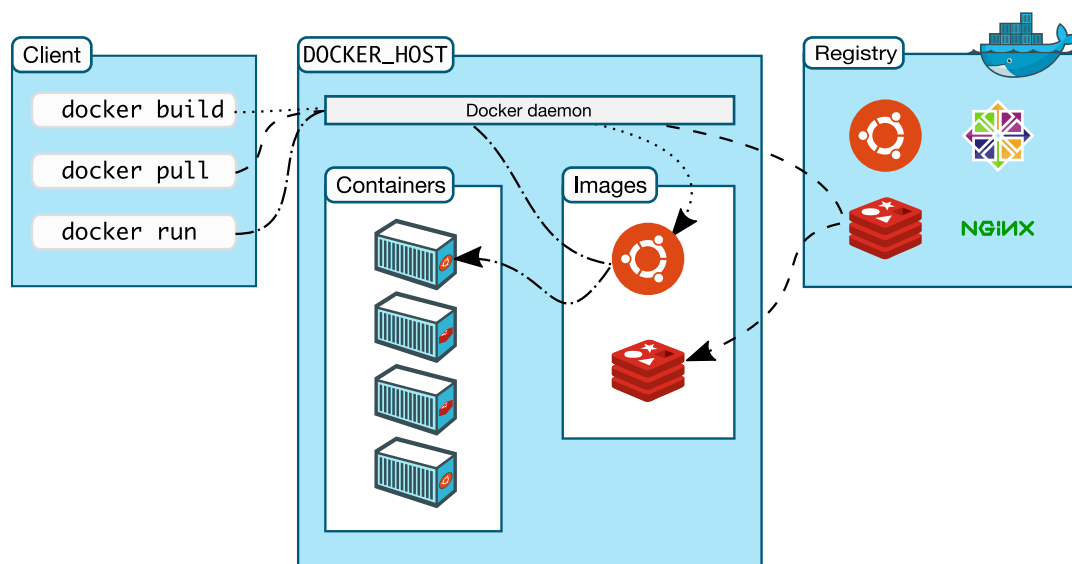


Figure 37.4: Docker Architecture

38. Running Docker Locally

The official installation documentation for docker can be found by visiting the following Web page:

- <https://www.docker.com/community-edition>

Here you will find a variety of packages, one of which will hopefully be suitable for your operating system. The supported operating systems currently include:

- OSX, Windows, CentOS, Debian, Fedora, Ubuntu, AWS, Azure

Please choose the one most suitable for you. For your convenience we provide you with installation instructions for OSX (Section 38.1), Windows 10 (Section 38.3) and Ubuntu (Section 38.2).

38.1 Installation for OSX

The docker community edition for OSX can be found at the following link

- <https://store.docker.com/editions/community/docker-ce-desktop-mac>

We recommend that at this time you get the version *Docker CE for MAC (stable)*

- <https://download.docker.com/mac/stable/Docker.dmg>

Clicking on the link will download a dmg file to your machine, that you then will need to install by double clicking and allowing access to the dmg file. Upon installation a whale in the top status bar shows that Docker is running, and you can access it via a terminal.



Figure 38.1: Docker integrated in the menu bar on OSX

38.2 Installation for Ubuntu

In order to instal Docker community edition for Ubuntu, you first have to register the repository from where you can download it. This can be achieved as follows:

```

1 $ sudo apt-get update
2 $ sudo apt-get install \
3     apt-transport-https \
4     ca-certificates \
5     curl \
6     software-properties-common
7 $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key <->
   ↪ add -
8 $ sudo apt-key fingerprint 0EBFCD88
9 $ sudo add-apt-repository \
10    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
11    $(lsb_release -cs) \
12    stable"

```

Now that you have configured the repository location, you can install it after you have updated the operating system. The update and install is done as follows:

```

1 $ sudo apt-get update
2 $ sudo apt-get install docker-ce
3 $ sudo apt-get update

```

38.3 Draft: Installation for Windows 10

Warning

The instructions for Windows have not passed our quality control. We look for TAs and student that test out and improve this section.

TODO: TA: Docker on windows. Please improve and finalize the section

Before we start we create the following directory:

```
1 mkdir $HOME/cloudmesh
```

We also assume you have installed gitbash and use the a git bash terminal window for the next steps.

To install Docker on Windows, please download the following files

- <https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe>
- <https://download.docker.com/win/stable/DockerToolbox.exe>

Move the downloaded files to a directory \$HOME/cloudmesh

Place the downloaded exe files in the cloudmesh directory we created earlier. First conduct the Docker installation and then continue with the Docker Toolbox installation. You can double click the exe files and run the installation.

When you are doing the installation, check on the selection in the installation saying to create shortcuts in your desktop. This way you can start all the programs from a desktop shortcut.

To run docker, you need to start the *Docker Quickstart Terminal* application and it will load all docker and provide a terminal window in which you can execute docker commands. Once the terminal is loaded, it will show something like following:

```
1 $ <username>@<yourpc> ~
```

38.4 Testing the Install

To test if it works execute the following commands in a terminal:

```
1 docker version
```

You should see an output similar to

```
1 docker version
2
3 Client:
4   Version:      17.03.1-ce
5   API version:  1.27
6   Go version:   go1.7.5
7   Git commit:   c6d412e
8   Built:        Tue Mar 28 00:40:02 2017
9   OS/Arch:      darwin/amd64
10
11 Server:
12  Version:      17.03.1-ce
13  API version:  1.27 (minimum version 1.12)
14  Go version:   go1.7.5
15  Git commit:   c6d412e
16  Built:        Fri Mar 24 00:00:50 2017
17  OS/Arch:      linux/amd64
18  Experimental: true
```

To see if you can run a container use

```
1 docker run hello-world
```

Once executed you should see an output similar to

```
1 Unable to find image 'hello-world:latest' locally
2 latest: Pulling from library/hello-world
3 78445dd45222: Pull complete
4 Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a .....
5 Status: Downloaded newer image for hello-world:latest
6
7 Hello from Docker!
8 This message shows that your installation appears to
9 be working correctly.
10
11 To generate this message, Docker took the following steps:
12 1. The Docker client contacted the Docker daemon.
13 2. The Docker daemon pulled the "hello-world" image
14    from the Docker Hub.
15 3. The Docker daemon created a new container from that
16    image which runs the executable that produces the
17    output you are currently reading.
```

```
18 4. The Docker daemon streamed that output to the Docker
19   client, which sent it to your terminal.
20
21 To try something more ambitious, you can run an Ubuntu container
22 with:
23
24 $ docker run -it ubuntu bash
25
26 Share images, automate workflows, and more with a free Docker ID:
27 https://cloud.docker.com/
28
29 For more examples and ideas, visit:
30 https://docs.docker.com/engine/userguide/
```




39. Docker and Docker Swarm on FutureSystems

Indiana University

This section is for IU students only that take classes with us.

This section introduces how to run Docker container on FutureSystems. Currently we have deployed Docker swarm on Echo.

39.1 Getting Access

You will need an account on FutureSystems and be enrolled in an active project. To verify, try to see if you can log into india.futuresystems.org. You need to be a member of a valid FutureSystems project, and had submitted an ssh public key via the FutureSystems portal.

Indiana University

For 2018 you need to be in the following project:
<https://portal.futuresystems.org/project/537>

If your access to the india host has been verified, try to login to the docker swarm head node. To conveniently do this let us define some Linux environment variables to simplify the access and the material presented here. YOU can place them even in your `.bashrc` or `.bash_profile` so the information gets populated whenever you start a new terminal.

```
1 export ECHO=149.165.150.76
2 export FS_USER=<put your futersystem here>
```

with the same username and key:

```
1 ssh $FS_USER@$ECHO
```

However it is much more convenient to

Note

If you have access to india but not the docker swarm system, your project may not have been authorized to access the docker swarm cluster. Send a ticket to FutureSystems ticket system to request this.

Once logged in to the docker swarm head node, try to run:

```
1 docker run hello-world
```

to verify `docker run` works.

39.2 Creating a service and deploy to the swarm cluster

While `docker run` can start a container and you may even attach to its console, the recommended way to use a docker swarm cluster is to create a service and have it run on the swarm cluster. The service will be scheduled to one or many number of the nodes of the swarm cluster, based on the configuration. It is also easy to scale up the service when more swarm nodes are available. Docker swarm really makes it easier for service/application developers to focus on the functionality development but not worrying about how and where to bind the service to some resources/server. The deployment, access, and scaling up/down when necessary, are all managed transparently. Thus achieving the new paradigm of *serverless computing*.

As an example, the following command creates a service and deploy it to the swarm cluster:

```
docker service create --name notebook_test -p 9001:8888 jupyter/datascience-notebook
start-notebook.sh --NotebookApp.password=NOTEBOOK_PASS_HASH
```

The `NOTEBOOK_PASS_HASH` can be generated in python:

```
1 >>> import IPython
2 >>> IPython.lib.passwd("YOUR_SELECTED_PASSWRD")
3 'sha1:52679cadb4c9:6762e266af44f86f3d170ca1.....'
```

So pass through the string starting with `'sha1:.....'`.

The command pulls a published image from docker cloud, starts a container and runs a script to start the service inside the container with necessary parameters. The option `“-p 9001:8888”` maps the service port inside the container (8888) to an external port of the cluster node (9001) so the service could be accessed from the Internet. In this example, you can then visit the URL:

[http://\\$ECHO:9001](http://$ECHO:9001)

to access the Jupyter notebook. Using the specified password when you create the service to login.

Please note the service will be dynamically deployed to a container instance, which would be allocated to a swarm node based on the allocation policy. Docker makes this process transparent to the user and even created mesh routing so you can access the service using the IP address of the management head node of the swarm cluster, no matter which actual physical node the service was deployed to.

This also implies that the external port number used has to be free at the time when the service was created.

Some useful related commands:

```
1 docker service ls
```

lists the currently running services.

```
1 docker service ps notebook_test
```

lists the detailed info of the container where the service is running.

```
1 docker node ps NODE
```

lists all the running containers of a node.

```
1 docker node ls
```

lists all the nodes in the swarm cluster.

To stop the service and the container:

```
1 docker service rm notebook_test
```

39.2.1 Create your own service

You can create your own service and run it. To do so, start from a base image, e.g., a ubuntu image from the docker cloud. Then you could:

- Run a container from the image and attach to its console to develop the service, and create a new image from the changed instance using command ‘docker commit’.
- Create a dockerfile, which has the step by step building process of the service, and then build an image from it.

In reality, the first approach is probably useful when you are in the phase of develop and debug your application/service. Once you have the step by step instructions developed the latter approach is the recommended way.

Publish the image to the docker cloud by following this documentation:

<https://docs.docker.com/docker-cloud/builds/push-images/>

Please make sure no sensitive information is included in the image to be published. Alternatively you could publish the image internally to the swarm cluster.

Publish an image privately within the swarm cluster

Once the image is published and available to the swarm cluster, you could start a new service from the image similar to the Jupyter Notebook example.

39.2.2 Exercises

Exercise 39.1 Obtain an account on future systems. ■

Exercise 39.2 Create a REST service with swagger codegen and run it on the echo cloud. ■

39.3 Dockerized REST Service

We discuss how to use Docker to deploy a REST service designed using Python Flask.

39.3.1 Prerequisites

In order to follow our discussion you will need a system on which you can run docker. This could either be OSX, Linux, or Windows.

Note

Python 2.7.x can be used to do this tutorial

Ubuntu and OSX

Please use our instructions on installing pyenv.

Windows

We start with download Python:

[Download Python MSI](#)

After installing python add an environmental variable by pressing Windows Key + Pause and Select Advanced system settings. Then add an environment variable for system variables for the variable PATH which is already there. And in that add the new variable

```
1 C:\Python27
```

As older python versions do not come or come with an outdated version of pip, we install it as follows:

[Download Pip Installer Script](#)

Now copy this file to

```
1 C:\Users\<<your_username>\cloudmesh\bin
```

If you do not have this path please create it, because we will be using this place to store all the tools we need. Within the bin folder run the following commands using command line tool or cmd.exe in windows.

```
1 $ python get-pip.py
```

Now add this environmental variable to PATH in System variables the same way we did earlier by putting the following value

```
1 C:\Python27\lib\site-packages
2 C:\Python27\Scripts
```

After adding the variables make sure you use a new cmd.exe.

Now continue by installing a Virtualenv installation

```
1 $ pip install virtualenv
```

Turn on Hyper-V (Windows Features Turn On and In the list select Hyper V)

Turn on Containers (Windows Features Turn On and In the list select Hyper V)

Install emacs via chocolatey

39.3.2 Activate Virtual Environment

Ubuntu and OSX

```

1 $ mkdir -p ~/cloudmesh/containers/docker-flask
2 $ cd ~/cloudmesh/containers/docker-flask
3 $ virtualenv venv
4 $ source venv/bin/activate

```

Windows

Using cmd.exe Please replace with your username.

```

1 $ mkdir -p C:\Users\\cloudmesh\containers\docker-flask
2 $ cd C:\Users\\cloudmesh\containers\docker-flask
3 $ virtualenv venv
4 $ venv/Script/activate

```

Now you are inside the created virtual environment. The terminal will look something like

```

1 (venv) neo$

```

39.3.3 File Structure

The File structure takes the following look.

```

1 docker-flask/[FOLDER]:[BASE PATH : ~/cloudmesh/containers/docker-flask]
2   |--|Dockerfile [FILE]
3   |--|requirements.txt [FILE]
4   |--|app/ [FOLDER]
5       |--|--|main.py [FILE]
6       |--|venv [FOLDER]

```

Step 1

Create requirements.txt file

Ubuntu and OSX

```

1 $ emacs requirements.txt

```

Windows

Install an editor such as emacs on Windows. YOU can use chocolatey for that, or use pycharm. Do not use notepad++

```

1 emacs requirements.txt

```

Include the following content in the requirements.txt file.

```

1 Flask==0.11.1

```

Now run the following command

```

1 $ pip install -r requirements.txt

```

Step 2

Then we are going to create the Dockerfile which includes all the instructions for the deployment of the REST API on docker.

Ubuntu and OSX

```
1 $ emacs Dockerfile
```

Windows

```
1 emacs Dockerfile
```

Include the following content in the Dockerfile

```
1 FROM tiangolo/uwsgi-nginx-flask:flask
2 COPY ./app /app
```

Here we have created a minimal Dockerfile.

FROM: command tells the image that has to be pulled from the Docker hub. So in this case for the ease of the task and to keep it simple with minimum packages we are going to select an image including nginx, flask and uWSGI.

1. Nginx: An open source software for web serving capability.
2. Flask: REST API package in Python
3. uWSGI: A server and one of the protocols implemented in WSGI which is a protocol used for pure HTTP communications and load balancing.

So this image includes everything we need to work on this tutorial.

COPY: Here it will copy content from app folder created in local machine to the Docker container.

Now the Dockerfile is completed and we have everything needed to build a docker image.

Step 3

Then we need to create the main.py file inside the app folder.

Ubuntu and OSX

```
1 emacs app/main.py
```

Windows

```
1 emacs app/main.py
```

Then add the following content.

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def hello():
7     return "Hey I am using Docker!"
8
9 @app.route("/api/cpu")
10 def get_cpu():
11     # ADD CODE TO GET CPU INFO
12     # USE psutil LIBRARY
13     return "SHOW YOUR CPU INFO"
14
15 if __name__ == "__main__":
16     app.run(host="127.0.0.1", debug=True, port=80)
```

After adding the content save and exit emacs.

39.3.4 Build Docker Image

Ubuntu and OSX

Now run the following commands.

```
1 $ cd ~/cloudmesh/containers/docker-flask
2 $ docker build -t sample-flask-rest-app .
```

Windows

Run Powershell as administrator and replace with your username.

```
1 cd C:\Users\\cloudmesh\containers\docker-flask
2 docker build -t sample-flask-rest-app .
```

If it builds successfully, you will get the following response

```
1 $ docker build -t sample-flask-rest-app .
2 Sending build context to Docker daemon 19.15MB
3 Step 1/2: FROM tiangolo/uwsgi-nginx-flask:python2.7
4 ---> ec43f17def9a
5 Step 2/2: COPY ./app /app
6 ---> e8eb1bff86b8
7 Successfully built e8eb1bff86b8
8 Successfully tagged sample-flask-rest-app:latest
9
10 Note: Changing any content inside the app folder must be
11 updated in the container by rebuilding the image.
```

39.3.5 Run Docker Image

Run the following commands to get the REST API hosted on <http://127.0.0.1:80>

Ubuntu and OSX

```
1 $ docker run -p 80:80 -t sample-flask-rest-app
```

Windows

In Windows powershell Run as administrator (use the previous powershell window)

```
1 $ docker run -p 80:80 -t sample-flask-rest-app
```

Here you may have to grant permission for using this port number. So please allow that to run.

It will take sometime to load the server once.

```
1 $ docker run -p 80:80 -t sample-flask-rest-app
```

If it is loaded and if it runs successfully you will see a response similar to

```
1 Checking for script in /app/prestart.sh
2 Running script /app/prestart.sh
3 Running inside /app/prestart.sh, you could add migrations to this file, e.g↔
↔ .:
4
```

```

5 #!/usr/bin/env bash
6
7 # Let the DB start
8 sleep 10;
9 # Run migrations
10 alembic upgrade head
11
12 /usr/lib/python2.7/dist-packages/supervisor/options.py:296:
13 UserWarning: Supervisor is running as root and it is searching
14 for its configuration file in default locations (including its
15 current working directory); you probably want to specify a "-c"
16 argument specifying an absolute path to a configuration file for
17 improved security.
18
19 'Supervisord is running as root and it is searching '
20 2018-02-19 18:07:46,198 CRIT Supervisor running as root
21     (no user in config file)
22 2018-02-19 18:07:46,198 WARN Included extra file
23     "/etc/supervisor/conf.d/supervisord.conf" during parsing
24 2018-02-19 18:07:46,204 INFO RPC interface 'supervisor' initialized
25 2018-02-19 18:07:46,204 CRIT Server 'unix_http_server'
26     running without any
27     HTTP authentication checking
28 2018-02-19 18:07:46,204 INFO supervisord started with pid 7
29 2018-02-19 18:07:47,207 INFO spawned: 'nginx' with pid 10
30 2018-02-19 18:07:47,211 INFO spawned: 'uwsgi' with pid 11
31 [uWSGI] getting INI configuration from /app/uwsgi.ini
32 [uWSGI] getting INI configuration from /etc/uwsgi/uwsgi.ini
33 *** Starting uWSGI 2.0.15 (64bit) on [Mon Feb 19 18:07:47 2018] ***
34 compiled with version: 4.9.2 on 04 February 2018 16:11:35
35 os: Linux-4.4.0-112-generic #135-Ubuntu SMP
36     Fri Jan 19 11:48:36 UTC 2018
37 nodename: 7f9706084219
38 machine: x86_64
39 clock source: unix
40 pcre jit disabled
41 detected number of CPU cores: 8
42 current working directory: /app
43 detected binary path: /usr/local/bin/uwsgi
44 your memory page size is 4096 bytes
45 detected max file descriptor number: 1048576
46 lock engine: pthread robust mutexes
47 thunder lock: disabled (you can enable it with --thunder-lock)
48 uwsgi socket 0 bound to UNIX address /tmp/uwsgi.sock fd 3
49 uWSGI running as root, you can use --uid/--gid/--chroot options
50 *** WARNING: you are running uWSGI as root !!!
51     (use the --uid flag) ***
52 Python version: 2.7.14 (default, Dec 12 2017, 16:55:09) [GCC 4.9.2]
53 *** Python threads support is disabled. You can enable it
54     with --enable-threads ***
55 Python main interpreter initialized at 0x1eed1b0
56 your server socket listen backlog is limited to 100 connections
57 your mercy for graceful operations on workers is 60 seconds
58 mapped 1237056 bytes (1208 KB) for 16 cores
59 *** Operational MODE: preforking ***
60 WSGI app 0 (mountpoint='') ready in 0 seconds on interpreter

```



```

61      Ox1eed1b0 pid: 11 (default app)
62 *** uWSGI is running in multiple interpreter mode ***
63 spawned uWSGI master process (pid: 11)
64 spawned uWSGI worker 1 (pid: 14, cores: 1)
65 spawned uWSGI worker 2 (pid: 15, cores: 1)
66 2018-02-19 18:07:48,357 INFO success: nginx entered RUNNING
67 state, process has stayed up for > than 1 seconds (startsecs)
68 2018-02-19 18:07:48,358 INFO success: uwsgi entered RUNNING
69 state, process has stayed up for > than 1 seconds (startsecs)

```

Go to this URL: <http://127.0.0.1:80>

Additional INFO

```

1 $ docker ps -a
2 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3
4 dc8cccf22216 35ffca69dcc3 "/entrypoint.sh /sta..." 4m ago Up 4m 443/tcp romantic_sammet
5 e7a45c81b237 sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago silly_kare
6 a4b6419016af sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago musing_lampport
7 6eb7102a514e prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago competent_borg
8 f7c6a4710ad2 prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago static-site
9 361c8812ba90 busybox "echo 'hello from bu..." 5d ago Exited (0) 5d ago blissful_wing
10 350ec9a2609f busybox "sh" 5d ago Exited (0) 5d ago nifty_mahavira
11 893cb11019f9 hello-world "/hello" 5d ago Exited (0) 5d ago competent_spence
12 1f90a411c746 hello-world "/hello" 11d ago Exited (0) 11d ago reverent_raman
13
14 $ docker stop dc8cccf22216
15 dc8cccf22216
16
17 $ docker ps -a
18 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
19 dc8cccf22216 35ffca69dcc3 "/entrypoint.sh /sta..." 4m ago Exited (137) 5s ago romantic_sammet
20 e7a45c81b237 sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago silly_kare
21 a4b6419016af sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago musing_lampport
22 6eb7102a514e prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago competent_borg
23 f7c6a4710ad2 prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago tatic-site
24 361c8812ba90 busybox "echo 'hello from bu..." 5d ago Exited (0) 5d ago blissful_wing
25 350ec9a2609f busybox "sh" 5d ago Exited (0) 5d ago ifty_mahavira
26 893cb11019f9 hello-world "/hello" 5d ago Exited (0) 5d ago competent_spence
27 1f90a411c746 hello-world "/hello" 11d ago Exited (0) 11d ago everent_raman

```

Deleting Docker Container first and then remove Docker Image

```

1 $ docker images
2 REPOSITORY TAG IMAGE ID CREATED SIZE
3 sample-flask-rest-app latest 8b3246425402 8m ago 697MB
4 <none> <none> 35ffca69dcc3 10m ago 697MB
5 sample-flask latest b763c65ae11b 2d ago 709MB
6 my-awesome-nginx v3 56cb2d15e863 3d ago 16.8MB
7 tiangolo/uwsgi-nginx-flask flask 3ab637f17463 2 weeks ago 709MB
8 tiangolo/uwsgi-nginx-flask python2.7 ec43f17def9a 2 weeks ago 697MB
9 ubuntu 16.04 0458a4468cbc 3 weeks ago 112MB
10 ubuntu latest 0458a4468cbc 3 weeks ago 112MB
11 busybox latest 5b0d59026729 3 weeks ago 1.15MB
12 nginx alpine bb00c21b4edf 5 weeks ago 16.8MB
13 hello-world latest f2a91732366c 3 months ago 1.85kB
14 prakhar1989/static-site latest f01030e1dcf3 2 years ago 134MB

```

```

1 $ docker ps -a
2 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3 74b9b994c9bd sample-flask-rest-app "/entrypoint.sh /sta..." 2m ago Exited (137) 1m ago infallible_mahavira
4 dc8cccf22216 35ffca69dcc3 "/entrypoint.sh /sta..." 10m ago Exited (137) 5m ago romantic_sammet
5 e7a45c81b237 sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago silly_kare
6 a4b6419016af sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago musing_lampport
7 6eb7102a514e prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago competent_borg
8 f7c6a4710ad2 prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago static-site
9 361c8812ba90 busybox "echo 'hello from bu..." 5d ago Exited (0) 5d ago blissful_wing
10 350ec9a2609f busybox "sh" 5d ago Exited (0) 5d ago nifty_mahavira
11 893cb11019f9 hello-world "/hello" 5d ago Exited (0) 5d ago competent_spence
12 1f90a411c746 hello-world "/hello" 11d ago Exited (0) 11d ago reverent_raman

```

```

1 $ docker rm 74b9b994c9bd
2 74b9b994c9bd

```

```

1 $ docker ps -a
2 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3 dc8cccf22216 35ffca69dcc3 "/entrypoint.sh /sta..." 10m ago Exited (137) 5m ago romantic_sammet
4 e7a45c81b237 sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago silly_kare
5 a4b6419016af sample-flask "/entrypoint.sh /usr..." 2d ago Exited (0) 2d ago musing_lampport
6 6eb7102a514e prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago competent_borg

```

```

7 f7c6a4710ad2 prakhar1989/static-site "./wrapper.sh" 5d ago Exited (0) 5d ago static-site
8 361c8812ba90 busybox "echo 'hello from bu..." 5d ago Exited (0) 5d ago blissful_wing
9 350ec9a2609f busybox "sh" 5d ago Exited (0) 5d ago nifty_mahavira
10 893cb11019f9 hello-world "/hello" 5d ago Exited (0) 5d ago competent_spence
11 1f90a411c746 hello-world "/hello" 11d ago Exited (0) 11d ago reverent_raman

```

```

1 $ docker images
2 REPOSITORY TAG IMAGE ID CREATED SIZE
3 sample-flask-rest-app latest 8b3246425402 8m ago 697MB
4 <none> <none> 35ffca69dcc3 10m ago 697MB
5 sample-flask latest b763c65ae11b 2d ago 709MB
6 my-awesome-nginx v3 56cb2d15e863 2d ago 16.8MB
7 tiangolo/uwsgi-nginx-flask flask 3ab637f17463 2 weeks ago 709MB
8 tiangolo/uwsgi-nginx-flask python2.7 ec43f17def9a 2 weeks ago 697MB
9 ubuntu 16.04 0458a4468cbc 3 weeks ago 112MB
10 ubuntu latest 0458a4468cbc 3 weeks ago 112MB
11 busybox latest 5b0d59026729 3 weeks ago 1.15MB
12 nginx alpine bb00c21b4edf 5 weeks ago 16.8MB
13 hello-world latest f2a91732366c 3 months ago 1.85kB
14 prakhar1989/static-site latest f01030e1dcf3 2 years ago 134MB

```

```

1 $ docker rmi 8b3246425402
2 Untagged: sample-flask-rest-app:latest
3 Deleted: sha256:8b3246425402b55aa5c4cf02cc5ad9ebd880b9fef639529b81495e778e3b3246
4 Deleted: sha256:639497d8f8bfa7cf497dfc142c8cc9d9b0b6b8689c777b9daa185c618b33d03c

```



40. Introduction to Kubernetes

40.1 Topics Covered and Learning Outcome

- What is Kubernetes?
- What are containers?
- Cluster components in Kubernetes
- Basic Units in Kubernetes
- Run an example with Minikube
- Interactive online tutorial
- Have a solid understanding of Containers and Kubernetes
- Understand the CLuster components of Kubernetes
- Understand the terminology of Kubernetes
- Gain practical experience with kubernetes
- With minikube
- With an interactive online tutorial

40.2 What is Kubernetes?

Kubernetes is an open-source platform designed to automate deploying, scaling, and operating application containers. <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

With Kubernetes, you can:

- Deploy your applications quickly and predictably.
- Scale your applications on the fly.
- Roll out new features seamlessly.
- Limit hardware usage to required resources only.
- Run applications in public and private clouds.

Kubernetes is

- Portable: public, private, hybrid, multi-cloud
- Extensible: modular, pluggable, hookable, composable
- Self-healing: auto-placement, auto-restart, auto-replication, auto-scaling

40.3 What are containers?

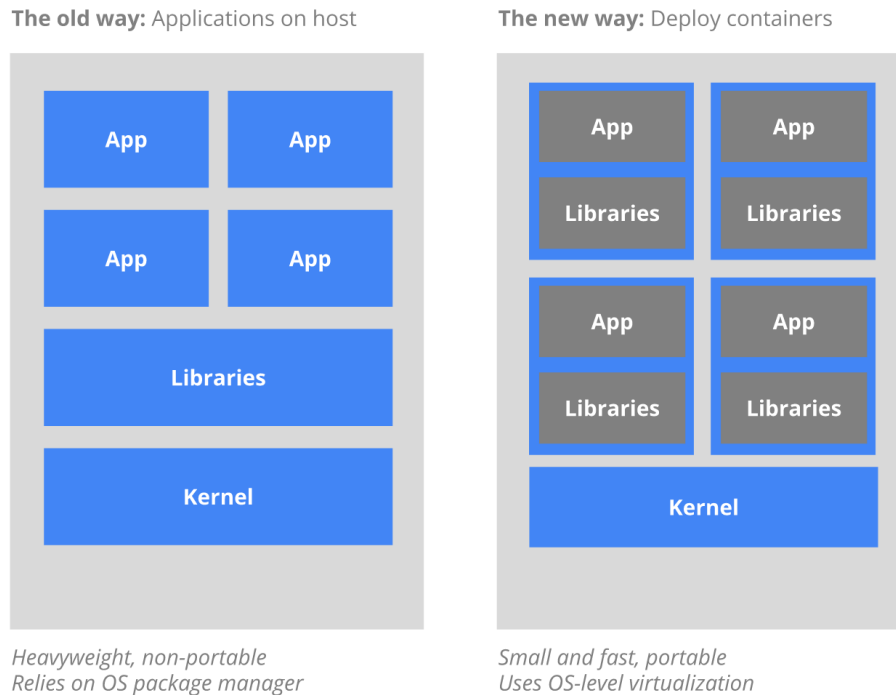


Figure 40.1: Kubernetes Containers

Image source:

- https://d33wubrfki0168.cloudfront.net/e7b766e0175f30ae37f7e0e349b87cfe2034a1ae/3e391/images/docs/why_containers.svg

40.4 Terminology

Pods: A pod (as in a pod of whales or pea pod) is a group of one or more containers (such as Docker containers), with shared storage/network, and a specification for how to run the containers. A pod's contents are always co-located and co-scheduled, and run in a shared context. A pod models an application-specific *logical host*. It contains one or more application containers which are relatively tightly coupled. In a pre-container world, they would have executed on the same physical or virtual machine.

Services: Service is an abstraction which defines a logical set of Pods and a policy by which to access them. Sometimes they are called a micro-service. The set of Pods targeted by a Service is (usually) determined by a Label Selector.

Deployments: A Deployment controller provides declarative updates for Pods and ReplicaSets. You describe a desired state in a Deployment object, and the Deployment controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

40.5 Kubernetes Architecture

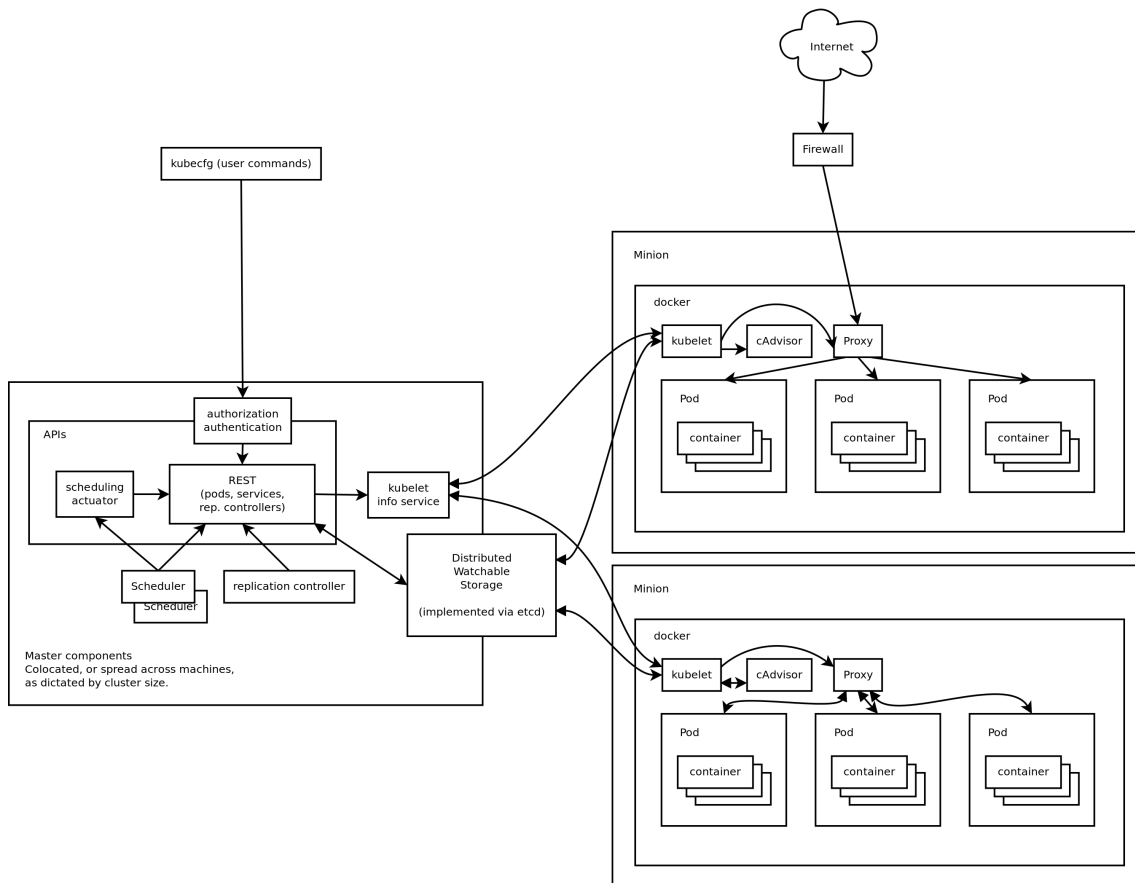


Figure 40.2: Kubernetes (Source: Google)

40.6 Minikube

1. minikube installation
2. minikube hello-minikube

40.6.1 Install minikube

OSX

```
1 curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/↔
   ↪ minikube-darwin-amd64 && chmod +x minikube && sudo mv minikube /usr/↔
   ↪ local/bin/
```

Windows 10

We assume that you have installed Oracle VirtualBox in your machine which must be a version 5.x.x.

Initially, we need to download two executables.

[Download Kubect1](#)

[Download Minikube](#)

After downloading these two executables place them in the cloudmesh directory we earlier created. Rename the minikube-windows-amd64.exe to minikube.exe. Make sure minikube.exe and kubectl.exe lie in the same directory.

Linux

```
1 curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/↵
   ↵ minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/↵
   ↵ local/bin/
```

Installing KVM2 is important for Ubuntu distributions

```
1 $ sudo apt install libvirt-bin qemu-kvm
2 $ sudo usermod -a -G libvirtd $(whoami)
3 $ newgrp libvirtd
```

We are going to run minikube using KVM2 libraries instead of virtualbox libraries for windows installation.

Then install the drivers for KVM2,

```
1 $ curl -LO https://storage.googleapis.com/minikube/releases/latest/docker-↵
   ↵ machine-driver-kvm2 && chmod +x docker-machine-driver-kvm2 && sudo ↵
   ↵ mv docker-machine-driver-kvm2 /usr/bin/
```

40.6.2 Start a cluster using Minikube

OSX Minikube Start

```
1 $ minikube start
```

Ubuntu Minikube Start

```
1 minikube start --vm-driver=kvm2
```

Windows 10 Minikube Start

In this case you must run Windows PowerShell as administrator. For this search for the application in search and right click and click Run as administrator. If you are an administrator it will run automatically but if you are not please make sure you provide the admin login information in the pop up.

```
1 $ cd C:\Users\\Documents\cloudmesh
2 $ .\minikube.exe start --vm-driver="virtualbox"
```

40.6.3 Create a deployment

```
1 $ kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --port=8080
```

40.6.4 Expose the service

```
1 $ kubectl expose deployment hello-minikube --type=NodePort
```

40.6.5 Check running status

This step is to make sure you have a pod up and running.

```
1 $ kubectl get pod
```

40.6.6 Call service api

```
1 $ curl $(minikube service hello-minikube --url)
```

40.6.7 Take a look from Dashboard

```
1 $ minikube dashboard
```

If you want to get an interactive dashboard,

```
1 $ .\minikube.exe dashboard --url=true
2 http://192.168.99.101:30000
```

Browse to <http://192.168.99.101:30000> in your web browser and it will provide a GUI dashboard regarding minikube.

40.6.8 Delete the service and deployment

```
1 $ kubectl delete service hello-minikube
2 $ kubectl delete deployment hello-minikube
```

40.6.9 Stop the cluster

For all platforms we can use the following command.

```
1 $ minikube stop
```

40.7 Interactive Tutorial Online

- Start cluster <https://kubernetes.io/docs/tutorials/kubernetes-basics/cluster-interactive/>
- Deploy app https://kubernetes.io/docs/tutorials/kubernetes-basics/cluster-interactive
- Explore <https://kubernetes.io/docs/tutorials/kubernetes-basics/explore-intro/>
- Expose <https://kubernetes.io/docs/tutorials/kubernetes-basics/expose-intro/>
- Scale <https://kubernetes.io/docs/tutorials/kubernetes-basics/scale-intro/>
- Update <https://kubernetes.io/docs/tutorials/kubernetes-basics/update-interactive/>
- MiniKube <https://kubernetes.io/docs/tutorials/stateless-application/hello-minikube/>

40.8 Using Kubernetes on FutureSystems

This section introduces you on how to use the Kubernetes cluster on FutureSystems. Currently we have deployed kubernetes on our cluster called *echo*.

40.8.1 Getting Access

You will need an account on FutureSystems and upload the ssh key to the FutureSystems portal from the computer from which you want to login to echo. To verify, if you have access try to see if you can log into india.futuresystems.org. You need to be a member of a valid FutureSystems project.

If you have verified that you have access to the india, you can now try to login to the kubernetes cluster head node with the same username and key:

At this time we ask you to use the following IP address to communicate with echo via its head node:

```
1 149.165.150.85
```

To login to echo use the command

```
1 ssh FS_USERNAME@149.165.150.85
```

where FS_USERUSER is the username you have on futureSystems.

NOTE: If you have access to india but not the kubernetes software, your project may not have been authorized to access the kubernetes cluster. Send a ticket to FutureSystems ticket system to request this.

Once you are logged in to the kubernetes cluster head node, try to run:

```
1 kubectl get pods
```

This will let you know if you have access to kubernetes and verifies if the kubectl command works for you. Naturally it will also list the pods.

40.8.2 Example Use

The following command runs an image called nginx with two replicas:

```
1 kubectl run nginx --replicas=2 --image=nginx --port=80
```

As a result of this one deployment was created, and two PODs are created and started. To see the deployment, please use the command

```
1 kubectl get deployment
```

This will result in the following output

1	NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
2	nginx	2	2	2	2	7m

To see the pods please use the command

```
1 kubectl get pods
```

This will result in the following output

1	NAME	READY	STATUS	RESTARTS	AGE
2	nginx-7587c6fdb6-4jnh6	1/1	Running	0	7m
3	nginx-7587c6fdb6-pxpsz	1/1	Running	0	7m

If we want to see more detailed information we can use the command

```
1 kubectl get pods -o wide
```

1	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
2	nginx-75...-4jnh6	1/1	Running	0	8m	192.168.56.2	e003
3	nginx-75...-pxpsz	1/1	Running	0	8m	192.168.255.66	e005

Please note the IP address field. Now if we try to access the nginx homepage with wget (or curl)

```
1 wget 192.168.56.2
```


we see the following output:

```

1 --2018-02-20 14:05:59-- http://192.168.56.2/
2 Connecting to 192.168.56.2:80... connected.
3 HTTP request sent, awaiting response... 200 OK
4 Length: 612 [text/html]
5 Saving to: 'index.html'
6
7 index.html 100%[=====>] 612 --.-KB/s in 0s
8
9 2018-02-20 14:05:59 (38.9 MB/s) - 'index.html' saved [612/612]
```

It verifies that the specified image was running, and it is accessible from within the cluster.

Next we need to start thinking about how we access this web server from outside the cluster. We can explicitly exposing the service with the following command

```
1 kubectl expose deployment nginx --type=NodePort --name=999-nginx-ext
```

We will see the response

```
1 service "nginx-external" exposed
```

To find the exposed ip addresses, we simply issue the command

```
1 kubectl get svc
```

We see something like this

NAME	TYPE	CLUSTER-IP	EXTERN AL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	8h
999-nginx-ext	NodePort	10.96.183.189	<none>	80:30275/TCP	6s

please note that we have given a unique name.

Indiana University

You could use your username or if you use one of our classes your hid. The number part will typically be sufficient. For class users that do not use the hid in the name we will terminate all instances without notification. In addition, we like you explicitly to add "-ext" to every container that is exposed to the internet. Naturally we want you to shut down such services if they are not in use. Failure to do so may result in termination of the service without notice, and in the worst case revocation of your priveledges to use *echo*.

In our example you will find the port on which our service is exposed and remaped to. We find the port **30275** in the value **80:30275/TCP** in the ports column for the running container.

Now if we visit this URL, which is the public IP of the head node followed by the exposed port number

```
1 http://149.165.150.85:30275
```

you should see the 'Welcome to nginx' page.

40.8.3 Exercises

Exercise 40.1 Explore more complex service examples. ■

Exercise 40.2 Explore constructing a complex web app with multiple services. ■

Exercise 40.3 Define a deployment with a yaml file declaratively. ■



41. Apache Hadoop using Docker

In this section we will explore the Map/Reduce framework using Hadoop provided through a Docker container. The example that we use in this session is similar to WordCount but simple calculations are added e.g. minimum, maximum, average and standard deviation values using several input files which contain float numbers.

41.1 Draft: Creating the Hadoop Container

41.2 Hadoop from Docker

Build a docker image by Dockerfile from:

```
1 mkdir hadoop
2 cd hadoop
3 wget https://raw.githubusercontent.com/cloudmesh/book/master/examples/↔
   ↪ docker/hadoop/Dockerfile
4 docker build -t cloudmesh/hadoop .
```

41.3 Start a Hadoop container

```
1 docker run -it cloudmesh/hadoop /etc/bootstrap.sh -bash
2 %docker run -it lee212/e222 /etc/bootstrap.sh -bash
```

It may take a few minutes at first to download image layers which are about 847MB.

41.4 Statistical Example with Hadoop

After a container is launched, the interactive shell prompt is given to run hadoop application which we have an example to get Min/Max/Avg/Std values by analyzing input text files.

41.4.1 Description

In a nutshell, this Hadoop program reads multiple files from HDFS and provides calculated values. We walk through every step from compiling Java source code to reading a output file from HDFS. The idea of this exercise is to get you started with Hadoop and the MapReduce concept. You may see the WordCount from Hadoop official website or documentation and this example has a same functions (Map/Reduce) except that you will be computing the basic statistics such as min, max, average, and standard deviation of a given data set.

The input to the program will be a text file(s) carrying exactly one floating point number per line. The result file includes *min*, *max*, *average*, and *standard deviation* of these numbers.

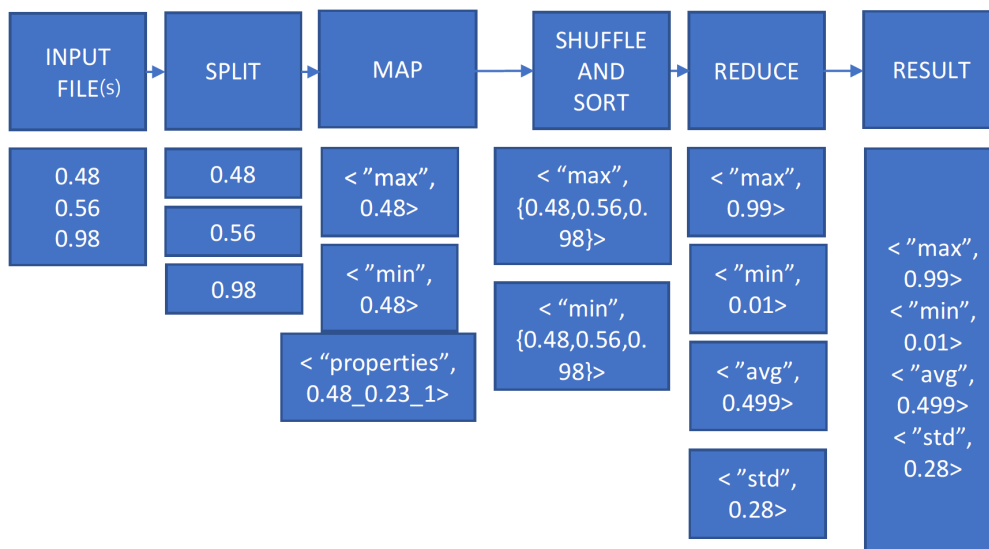


Figure 41.1: Caption Missing

41.4.2 Base Location

The example is available at:

```
1 cd /cloudmesh/exer1
```

41.4.3 Input Files

A test input files are available under `/cloudmesh/exer1/input_data` directory inside of the container. The statistics values for this input are *Min: 0.20 Max: 19.99 Avg: 9.51 StdDev: 5.55* for all input files.

10 files contain 55000 lines to process and each line is a random float point value ranging from 0.2 to 20.0.

41.4.4 Compilation

The source code file name is `MinMaxAvgStd.java` which is available at `/cloudmesh/exer1/src/exercise/`.

There are three functions in the code `Map`, `Reduce` and `Main` where `Map` reads each line of a file and updates values to calculate minimum, maximum values and `Reduce` collects mappers to produce average and standard deviation values at last.

```

1 export HADOOP_CLASSPATH='$HADOOP_PREFIX/bin/hadoop classpath'
2 mkdir /cloudmesh/exer1/dest
3 javac -classpath $HADOOP_CLASSPATH -d /cloudmesh/exer1/dest /cloudmesh/↔
↔ exer1/src/exercise/MinMaxAvgStd.java

```

These commands simply prepare compiling the example code and the compiled class files are generated at the *dest* location.

41.4.5 Archiving Class Files

Jar command tool helps archiving classes in a single file which will be used when Hadoop runs this example. This is useful because a jar file contains all necessary files to run a program.

```

1 cd /cloudmesh/exer1
2 jar -cvf exer1.jar -C ./dest/ .

```

41.4.6 HDFS for Input/Output

The input files need to be uploaded to HDFS as Hadoop runs this example by reading input files from HDFS.

```

1 export PATH=$PATH:$HADOOP_PREFIX/bin
2 hadoop fs -mkdir exer1_input
3 hadoop fs -put input_data/* exer1_input
4 hadoop fs -ls exer1_input/

```

If uploading is completed, you may see file listings like:

```

1 Found 10 items
2 -rw-r--r-- 1 root supergroup 13942 2018-02-28 23:16 exer1_input/data_1000.↔
↔ txt
3 -rw-r--r-- 1 root supergroup 139225 2018-02-28 23:16 exer1_input/data_10000↔
↔ .txt
4 -rw-r--r-- 1 root supergroup 27868 2018-02-28 23:16 exer1_input/data_2000.↔
↔ txt
5 -rw-r--r-- 1 root supergroup 41793 2018-02-28 23:16 exer1_input/data_3000.↔
↔ txt
6 -rw-r--r-- 1 root supergroup 55699 2018-02-28 23:16 exer1_input/data_4000.↔
↔ txt
7 -rw-r--r-- 1 root supergroup 69663 2018-02-28 23:16 exer1_input/data_5000.↔
↔ txt
8 -rw-r--r-- 1 root supergroup 83614 2018-02-28 23:16 exer1_input/data_6000.↔
↔ txt
9 -rw-r--r-- 1 root supergroup 97490 2018-02-28 23:16 exer1_input/data_7000.↔
↔ txt
10 -rw-r--r-- 1 root supergroup 111451 2018-02-28 23:16 exer1_input/data_8000.↔
↔ txt
11 -rw-r--r-- 1 root supergroup 125337 2018-02-28 23:16 exer1_input/data_9000.↔
↔ txt

```

41.4.7 Run Program with a Single Input File

We are ready to run the program to calculate values from text files. First, we simply run the program with a single input file to see how it works. `data_1000.txt` contains 1000 lines of floats, we use this file here.


```
1 hadoop jar exer1.jar exercise.MinMaxAvgStd exer1_input/data_1000.txt ↵
   ↵ exer1_output_1000
```

The command runs with input parameters which indicate a jar file (the program, `exer1.jar`), exercise.MinMaxAvgStd (package name.class name), input file path (`exer1_input/data_1000.txt`) and output file path (`exer1_output_1000`).

The sample results that the program produces look like this:

```
1 18/02/28 23:48:50 INFO client.RMPProxy: Connecting to ResourceManager at ↵
   ↵ /0.0.0.0:8032
2 18/02/28 23:48:50 INFO input.FileInputFormat: Total input paths to process:↵
   ↵ 1
3 18/02/28 23:48:50 INFO mapreduce.JobSubmitter: number of splits:1
4 18/02/28 23:48:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: ↵
   ↵ job_1519877569596_0002
5 18/02/28 23:48:51 INFO impl.YarnClientImpl: Submitted application ↵
   ↵ application_1519877569596_0002
6 18/02/28 23:48:51 INFO mapreduce.Job: The url to track the job: http://↵
   ↵ f5e82d68ba4a:8088/proxy/application_1519877569596_0002/
7 18/02/28 23:48:51 INFO mapreduce.Job: Running job: job_1519877569596_0002
8 18/02/28 23:48:56 INFO mapreduce.Job: Job job_1519877569596_0002 running in↵
   ↵ uber mode: false
9 18/02/28 23:48:56 INFO mapreduce.Job: map 0% reduce 0%
10 18/02/28 23:49:00 INFO mapreduce.Job: map 100% reduce 0%
11 18/02/28 23:49:05 INFO mapreduce.Job: map 100% reduce 100%
12 18/02/28 23:49:05 INFO mapreduce.Job: Job job_1519877569596_0002 completed ↵
   ↵ successfully
13 18/02/28 23:49:05 INFO mapreduce.Job: Counters: 49
14   File System Counters
15     FILE: Number of bytes read=81789
16     FILE: Number of bytes written=394101
17     FILE: Number of read operations=0
18     FILE: Number of large read operations=0
19     FILE: Number of write operations=0
20     HDFS: Number of bytes read=14067
21     HDFS: Number of bytes written=86
22     HDFS: Number of read operations=6
23     HDFS: Number of large read operations=0
24     HDFS: Number of write operations=2
25   Job Counters
26     Launched map tasks=1
27     Launched reduce tasks=1
28     Data-local map tasks=1
29     Total time spent by all maps in occupied slots (ms)=2107
30     Total time spent by all reduces in occupied slots (ms)=2316
31     Total time spent by all map tasks (ms)=2107
32     Total time spent by all reduce tasks (ms)=2316
33     Total vcore-seconds taken by all map tasks=2107
34     Total vcore-seconds taken by all reduce tasks=2316
35     Total megabyte-seconds taken by all map tasks=2157568
36     Total megabyte-seconds taken by all reduce tasks=2371584
37   Map-Reduce Framework
38     Map input records=1000
39     Map output records=3000
40     Map output bytes=75783
41     Map output materialized bytes=81789
```

```

42   Input split bytes=125
43   Combine input records=0
44   Combine output records=0
45   Reduce input groups=3
46   Reduce shuffle bytes=81789
47   Reduce input records=3000
48   Reduce output records=4
49   Spilled Records=6000
50   Shuffled Maps =1
51   Failed Shuffles=0
52   Merged Map outputs=1
53   GC time elapsed (ms)=31
54   CPU time spent (ms)=1440
55   Physical memory (bytes) snapshot=434913280
56   Virtual memory (bytes) snapshot=1497260032
57   Total committed heap usage (bytes)=402653184
58   Shuffle Errors
59     BAD_ID=0
60     CONNECTION=0
61     IO_ERROR=0
62     WRONG_LENGTH=0
63     WRONG_MAP=0
64     WRONG_REDUCE=0
65   File Input Format Counters
66     Bytes Read=13942
67   File Output Format Counters
68     Bytes Written=86

```

The second line of the following logs indicates that the number of input files is 1.

41.4.8 Result for Single Input File

We reads results from HDFS by:

```
1 hadoop fs -cat exer1_output_1000/part-r-00000
```

The sample output looks like:

```

1 Max: 19.9678704297
2 Min: 0.218880718983
3 Avg: 10.225467263249385
4 Std: 5.679809322880863

```

41.4.9 Run Program with Multiple Input Files

The first run was done pretty quickly (1440 milliseconds took according to the sample result above) because the input file size was small (1,000 lines) and it was a single file. We provides more input files with a larger size (2,000 to 10,000 lines). Input files are already uploaded to HDFS. We simply run the program again with a slight change in the parameters.

```
1 hadoop jar exer1.jar exercise.MinMaxAvgStd exer1_input/ exer1_output_all
```

The command is almost same except that an input path is a directory and a new output directory. Note that every time that you run this program, the output directory will be created which means that you have to provide a new directory name unless you delete it.

The sample output messages look like the following which is almost identical compared to the previous run except that this time the number of input files to process is 10, see the line two below:

```

1 18/02/28 23:17:18 INFO client.RMPProxy: Connecting to ResourceManager at ↵
   ↵ /0.0.0.0:8032
2 18/02/28 23:17:18 INFO input.FileInputFormat: Total input paths to process:↵
   ↵ 10
3 18/02/28 23:17:18 INFO mapreduce.JobSubmitter: number of splits:10
4 18/02/28 23:17:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: ↵
   ↵ job_1519877569596_0001
5 18/02/28 23:17:19 INFO impl.YarnClientImpl: Submitted application ↵
   ↵ application_1519877569596_0001
6 18/02/28 23:17:19 INFO mapreduce.Job: The url to track the job: http://↵
   ↵ f5e82d68ba4a:8088/proxy/application_1519877569596_0001/
7 18/02/28 23:17:19 INFO mapreduce.Job: Running job: job_1519877569596_0001
8 18/02/28 23:17:24 INFO mapreduce.Job: Job job_1519877569596_0001 running in↵
   ↵ uber mode: false
9 18/02/28 23:17:24 INFO mapreduce.Job: map 0% reduce 0%
10 18/02/28 23:17:32 INFO mapreduce.Job: map 40% reduce 0%
11 18/02/28 23:17:33 INFO mapreduce.Job: map 60% reduce 0%
12 18/02/28 23:17:36 INFO mapreduce.Job: map 70% reduce 0%
13 18/02/28 23:17:37 INFO mapreduce.Job: map 100% reduce 0%
14 18/02/28 23:17:39 INFO mapreduce.Job: map 100% reduce 100%
15 18/02/28 23:17:39 INFO mapreduce.Job: Job job_1519877569596_0001 completed ↵
   ↵ successfully
16 18/02/28 23:17:39 INFO mapreduce.Job: Counters: 49
17   File System Counters
18     FILE: Number of bytes read=4496318
19     FILE: Number of bytes written=10260627
20     FILE: Number of read operations=0
21     FILE: Number of large read operations=0
22     FILE: Number of write operations=0
23     HDFS: Number of bytes read=767333
24     HDFS: Number of bytes written=84
25     HDFS: Number of read operations=33
26     HDFS: Number of large read operations=0
27     HDFS: Number of write operations=2
28   Job Counters
29     Launched map tasks=10
30     Launched reduce tasks=1
31     Data-local map tasks=10
32     Total time spent by all maps in occupied slots (ms)=50866
33     Total time spent by all reduces in occupied slots (ms)=4490
34     Total time spent by all map tasks (ms)=50866
35     Total time spent by all reduce tasks (ms)=4490
36     Total vcore-seconds taken by all map tasks=50866
37     Total vcore-seconds taken by all reduce tasks=4490
38     Total megabyte-seconds taken by all map tasks=52086784
39     Total megabyte-seconds taken by all reduce tasks=4597760
40   Map-Reduce Framework
41     Map input records=55000
42     Map output records=165000
43     Map output bytes=4166312
44     Map output materialized bytes=4496372
45     Input split bytes=1251
46     Combine input records=0

```



```
47   Combine output records=0
48   Reduce input groups=3
49   Reduce shuffle bytes=4496372
50   Reduce input records=165000
51   Reduce output records=4
52   Spilled Records=330000
53   Shuffled Maps =10
54   Failed Shuffles=0
55   Merged Map outputs=10
56   GC time elapsed (ms)=555
57   CPU time spent (ms)=16040
58   Physical memory (bytes) snapshot=2837708800
59   Virtual memory (bytes) snapshot=8200089600
60   Total committed heap usage (bytes)=2213019648
61   Shuffle Errors
62     BAD_ID=0
63     CONNECTION=0
64     IO_ERROR=0
65     WRONG_LENGTH=0
66     WRONG_MAP=0
67     WRONG_REDUCE=0
68   File Input Format Counters
69     Bytes Read=766082
70   File Output Format Counters
71     Bytes Written=84
```

41.4.10 Result for Multiple Files

```
1 hadoop fs -cat exer1_output_all/part-r-00000
```

The expected result looks like:

```
1 Max: 19.999191254
2 Min: 0.200268613863
3 Avg: 9.514884854468903
4 Std: 5.553921579413547
```

41.5 Conclusion

The example program of calculating some values by reading multiple files shows how Map/Reduce is written by a Java programming language and how Hadoop runs its program using HDFS. We also observed the one of benefits using Docker container which is that the hassle of configuration and installation of Hadoop is not necessary anymore.

A large container ship is shown from a low angle, sailing on the water. The ship's deck is filled with stacks of colorful shipping containers in shades of green, blue, and red. The sky is a soft, hazy orange, suggesting a sunset or sunrise. A blue-bordered white box is overlaid on the lower part of the image, containing the chapter title.

42. Apache Hadoop Latest (3.0.1) using Docker

Apache Hadoop 3.0.1 is the latest release (25 March, 2018) which includes significant enhancements over the previous version of Hadoop 2.x.

This section provides a separated Dockerfile to build hadoop 3.0.1 but the configurations and examples are same except a few minor changes which are:

- CentOS 7 - systemctl - Java SE Development Kit 8

42.1 Draft: Building Hadoop 3.0.1 using Docker

Build a docker image by Dockerfile from:

```
1 mkdir hadoop3.0.1
2 cd hadoop3.0.1
3 wget https://raw.githubusercontent.com/cloudmesh/book/master/examples/↵
   ↪ docker/hadoop/3.0.1/Dockerfile
4 docker build -t cloudmesh/hadoop:3.0.1 .
```

The complete docker image for Hadoop 3.0.1 consumes the storage size of 1.5GB.

```
1 \$$ docker images
2 REPOSITORY          TAG          IMAGE ID          ↵
   ↪ CREATED          SIZE
3 cloudmesh/hadoop    3.0.1       ba2c51f94348     20 ↵
   ↪ hours ago        1.5GB
```

42.2 Start a Hadoop Container with Interactive Shell

```
1 docker run -it cloudmesh/hadoop:3.0.1 /etc/bootstrap.sh -bash
2 %docker run -it lee212/e222 /etc/bootstrap.sh -bash
```

The details of the new version is available from the official site here: <http://hadoop.apache.org/docs/r3.0.1/index.html>

42.3 Examples

The statistics and PageRank examples are identical to the Hadoop 2.7.5 on Docker [40.8.3](#).

42.4 Draft: Apache Spark with Docker

42.4.1 Pull Image from Docker Repository

We use a Docker image from Docker Hub: (<https://hub.docker.com/r/sequenceiq/spark/>) This repository contains a Docker file to build a Docker image with Apache Spark and Hadoop Yarn.

```
1 docker pull sequenceiq/spark:1.6.0
```

42.4.2 Running the Image

In this step, we will launch a Spark container.

Running interactively

```
1 docker run -it -p 8088:8088 -p 8042:8042 -h sandbox sequenceiq/spark:1.6.0 ↵
  ↵ bash
```

Running in the background

```
1 docker run -d -h sandbox sequenceiq/spark:1.6.0 -d
```

42.4.3 Run Spark

After a container is launched, we can run Spark in the following two modes: (1) yarn-client and (2) yarn-cluster. The differences between the two modes can be found here: <https://spark.apache.org/docs/latest/running-on-yarn.html>

Run Spark in Yarn-Client Mode

```
1 spark-shell --master yarn-client --driver-memory 1g --executor-memory 1g --↵
  ↵ executor-cores 1
```

Run Spark in Yarn-Cluster Mode

```
1 spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client↵
  ↵ --driver-memory 1g --executor-memory 1g --executor-cores 1 ↵
  ↵ $SPARK_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar
```

42.4.4 Observe Task Execution from Running Logs of SparkPi

Let us observe Spark task execution by adjusting the parameter of SparkPi and the Pi result from the following two commands.

```
1 spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client↵
  ↵ --driver-memory 1g --executor-memory 1g --executor-cores 1 ↵
  ↵ $SPARK_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar 10
```

```
1 spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client↵
  ↵ --driver-memory 1g --executor-memory 1g --executor-cores 1 ↵
  ↵ $SPARK_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar 10000
```

42.4.5 Write a Word-Count Application with Spark RDD

Let us write our own word-count with Spark RDD. After the shell has been started, copy and paste the following code in console line by line.

Launch Spark Interactive Shell

```
1 spark-shell --master yarn-client --driver-memory 1g --executor-memory 1g --↵
  ↵ executor-cores 1
```

Program in Scala

```
1 val textFile = sc.textFile("file:///etc/hosts")
2 val words = textFile.flatMap(line => line.split("\\s+"))
3 val counts = words.map(word => (word, 1)).reduceByKey(_ + _)
4 counts.values.sum()
```

Launch PySpark Interactive Shell

```
1 pyspark --master yarn-client --driver-memory 1g --executor-memory 1g --↵
  ↵ executor-cores 1
```

Program in Python

```
1 textFile = sc.textFile("file:///etc/hosts")
2 words = textFile.flatMap(lambda line:line.split())
3 counts = words.map(lambda word:(word, 1)).reduceByKey(lambda x,y: x+y)
4 counts.map(lambda x:x[1]).sum()
```

42.4.6 Docker Spark Examples

K-Means Example

First we need to pull the image from the Docker Hub :

```
1 docker pull sequenceiq/spark-native-yarn
```

It will take sometime to download the image. Now we have to run docker spark image interactively.

```
1 docker run -i -t -h sandbox sequenceiq/spark-native-yarn /etc/bootstrap.sh ↵
  ↵ -bash
```

This will take you to the interactive mode.

Let's run a sample KMeans example. This is already built with Spark.

Here we specify the data data set from a local folder inside the image and we run the sample class KMeans in the sample package. The sample data set used is inside the sample-data folder. Spark has it's own format for machine learning datasets. Here the kmeans_data.txt file contains the KMeans dataset.

```
1 ./bin/spark-submit --class sample.KMeans --master execution-context:org.↵
  ↵ apache.spark.tez.TezJobExecutionContext --conf update-classpath=true↵
  ↵ ./lib/spark-native-yarn-samples-1.0.jar /sample-data/kmeans_data.↵
  ↵ txt
```

If you run this successfully, you can get an output as shown here.

```

1 Finished iteration (delta = 0.0)
2 Final centers:
3 DenseVector(0.15000000000000002, 0.15000000000000002, 0.15000000000000002)
4 DenseVector(9.2, 9.2, 9.2)
5 DenseVector(0.0, 0.0, 0.0)
6 DenseVector(9.05, 9.05, 9.05)

```

Join Example

Run the following command to do a sample join operation on a given dataset. Here we use two datasets, namely join1.txt and join2.txt. Then we perform the join operation that we discussed in the theory section.

```

1 ./bin/spark-submit --class sample.Join --master execution-context:org.↵
  ↵ apache.spark.tez.TezJobExecutionContext --conf update-classpath=true↵
  ↵ ./lib/spark-native-yarn-samples-1.0.jar /sample-data/join1.txt /↵
  ↵ sample-data/join2.txt

```

Word Count

In this example the wordcount.txt will used to do the word count using multiple reducers. Number 1 at the end of the command determines the number of reducers. As spark can run multiple reducers, we can specify the number as a parameter to the programme.

```

1 ./bin/spark-submit --class sample.WordCount --master execution-context:org.↵
  ↵ apache.spark.tez.TezJobExecutionContext --conf update-classpath=true↵
  ↵ ./lib/spark-native-yarn-samples-1.0.jar /sample-data/wordcount.txt ↵
  ↵ 1

```

42.4.7 Interactive Examples

Here we need a new image to work on. Let's run the following command. This will pull the necessary repositories from docker hub, as we don't have most of the dependencies related to it. This can take a few minutes to download everything.

```

1 docker run -it-p 8888:8888 -v $PWD:/cloudmesh/spark --name spark jupyter/↵
  ↵ pyspark-notebook

```

Here you will get the following output in the terminal.

```

vibhatha@vibhatha-ThinkPad-P50:~/Sandbox/docker/spark$ docker run -it -p 8888:8888 -v $PWD:/home/cloudmesh/work --name spark jupyter/pyspark-notebook
Container must be run with group root to update passwd file
Executing the command: jupyter notebook
[I 17:34:20.028 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[W 17:34:20.320 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption. This is not recommended.
[I 17:34:20.350 NotebookApp] JupyterLab beta preview extension loaded from /opt/conda/lib/python3.6/site-packages/jupyterlab
[I 17:34:20.350 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 17:34:20.358 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 17:34:20.358 NotebookApp] 0 active kernels
[I 17:34:20.358 NotebookApp] The Jupyter Notebook is running at:
[I 17:34:20.358 NotebookApp] http://[all ip addresses on your system]:8888/?token=dadb9c035a5f9a12bbaa385f3e80c18f71f0a734fe576c04
[I 17:34:20.358 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:34:20.359 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
  http://localhost:8888/?token=dadb9c035a5f9a12bbaa385f3e80c18f71f0a734fe576c04

```

Figure 42.1: Terminal Output

Please copy the url shown at the end of the terminal output and go to that url in the browser.

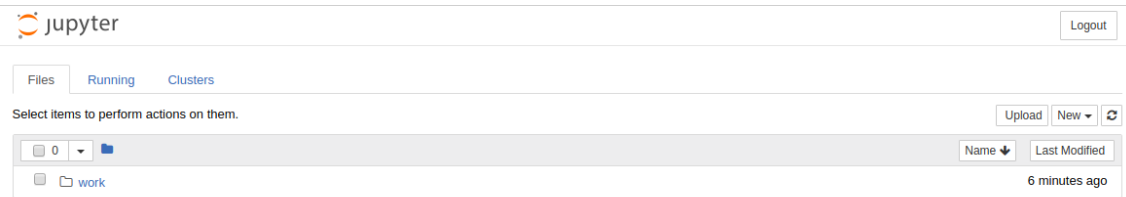


Figure 42.2: Jupyter Notebook in Browser

You will see the following output in the browser, (Use Google Chrome)

First navigate to the work folder. Let us create a new python file here. Click python3 in the new menu.



Figure 42.3: Create a new python file

Now add the following content in the new file. In Jupyter notebook, you can enter a python command or python code and press

```
1 SHIFT + ENTER
```

This will run the code interactively.

Now let's create the following content.

Now let us do the following.

In the following stage we configure spark context and import the necessary files.

Next stage we use sample data set by creating them in form of an array and we train the kmeans algorithm.

In the final stage we put sample values and check the predictions on the cluster. In addition to that feed the data using SparseVector format and we add the kmeans initialization mode, the error margin and the parallelization. We put the step size as 5 for this example. In the previous one we didn't specify any parameters.

The predict term predicts the cluster id which it belongs to.

Then in the following way you can check whether two data points belong to one cluster or not.

Stop Docker Container

```
1 docker stop spark
```

Start Docker Container Again

```
1 docker start spark
```

The screenshot shows a Jupyter Notebook with the following content:

```

In [1]: import os
os.getcwd()
# owner of the pyspark notebook created the working directory as follows.

Out[1]: '/home/jovyan/work'

In [2]: import pyspark
sc = pyspark.SparkContext('local[*]')
# do something to prove it works
rdd = sc.parallelize(range(1000))
rdd.takeSample(False, 5)

Out[2]: [101, 59, 336, 889, 19]

```

Figure 42.4: Create a new python file

The screenshot shows a Jupyter Notebook with the following content:

```

In [15]: # runs a simple exercise on rdd ( 5 samples picked up out of 1000)
import pyspark
sc = pyspark.SparkContext()
# this can only be run once (because multiple Spark Contexts won't be allowed)

In [19]: rdd = sc.parallelize(range(1000))
rdd.takeSample(False, 5)
# run this multiple times and you will get different values as the output

Out[19]: [623, 304, 442, 329, 992]

In [6]: # make data folder
os.makedirs("data")
# running this again will give an error as this folder already exists

```

Figure 42.5: Initial Spark Program


```

In [16]: sparse_data = [
...     SparseVector(3, {1: 1.0}),
...     SparseVector(3, {1: 1.1}),
...     SparseVector(3, {2: 1.0}),
...     SparseVector(3, {2: 1.1})
... ]

In [29]: model = KMeans.train(sc.parallelize(sparse_data), 2, initializationMode="k-means||",
...                               seed=50, initializationSteps=5, epsilon=1e-4)

In [30]: model.predict(array([0., 1., 0.]))

Out[30]: 0

In [31]: model.predict(array([0., 0., 1.]))

Out[31]: 1

In [32]: model.predict(sparse_data[0])

Out[32]: 0

In [33]: model.predict(sparse_data[2])

Out[33]: 1

```

Figure 42.6: Train KMeans

```

In [9]: data = array([0.0,0.0, 1.0,1.0, 9.0,8.0, 8.0,9.0]).reshape(4, 2)

In [26]: model = KMeans.train(
...     sc.parallelize(data), 2, maxIterations=10, initializationMode="random",
...     seed=50, initializationSteps=5, epsilon=1e-4)

In [27]: model.predict(array([0.0, 0.0])) == model.predict(array([1.0, 1.0]))

Out[27]: True

In [28]: model.predict(array([8.0, 9.0]))

Out[28]: 1

In [12]: model.predict(array([8.0, 9.0])) == model.predict(array([9.0, 8.0]))

Out[12]: True

In [13]: model.k

Out[13]: 2

In [14]: model.computeCost(sc.parallelize(data))

Out[14]: 2.0000000000000004

In [15]: model = KMeans.train(sc.parallelize(data), 2)

```

Figure 42.7: Predict KMeans Clusters-1

```

In [41]: data = array([0.0,0.0, 1.0,1.0, 9.0,8.0, 8.0,9.0]).reshape(4, 2)

In [42]: model = KMeans.train(
...     sc.parallelize(data), 2, maxIterations=10, initializationMode="random",
...     seed=50, initializationSteps=5, epsilon=1e-4)

In [43]: model.predict(array([0.0, 0.0])) == model.predict(array([1.0, 1.0]))
Out[43]: True

In [44]: model.predict(array([8.0, 9.0])) == model.predict(array([9.0, 8.0]))
Out[44]: True

In [45]: model.k
Out[45]: 2

In [46]: model.computeCost(sc.parallelize(data))
Out[46]: 2.0000000000000004

In [47]: model = KMeans.train(sc.parallelize(data), 2)

```

Figure 42.8: Predict KMeans Clusters-2

```

In [53]: sparse_data = [
...     SparseVector(3, {1: 1.0}),
...     SparseVector(3, {1: 1.1}),
...     SparseVector(3, {2: 1.0}),
...     SparseVector(3, {2: 1.1})
... ]

In [54]: model = KMeans.train(sc.parallelize(sparse_data), 2, initializationMode="k-means||",
...     seed=50, initializationSteps=5, epsilon=1e-4)

In [55]: model.predict(array([0., 1., 0.])) == model.predict(array([0, 1.1, 0.]))
True
Out[55]: True

In [56]: model.predict(array([0., 0., 1.])) == model.predict(array([0, 0, 1.1]))
Out[56]: True

In [57]: model.predict(sparse_data[0]) == model.predict(sparse_data[1])
Out[57]: True

In [58]: model.predict(sparse_data[2]) == model.predict(sparse_data[3])
Out[58]: True

In [59]: isinstance(model.clusterCenters, list)
Out[59]: True

```

Figure 42.9: Predict KMeans Clusters-3

Remove Docker Container

```
1 docker rm spark
```

42.5 Draft:Multinode Hadoop Cluster Deployment with Docker Swarm

42.5.1 Docker Hadoop 3.0.1

This Dockerfile builds Hadoop Docker container with CentOS base image and contains examples to run Hadoop such as statistics like WordCount and PageRank.

Build

```
docker build -t cloudmesh/hadoop:3.0.1 .
```

Run

```
docker run -it cloudmesh/hadoop:3.0.1 /etc/bootstrap.sh -bash
```

PageRank Example

Find instruction and source code to run in the following directory:

```
/cloudmesh/pagerank
```

Hadoop Configuration Files

The configuration files are shared with Hadoop 2.7.5.

Virtual Memory Limit

Increase memory limit in the mapred-site.xml, for example:

- mapreduce.map.memory.mba to 4096
- mapreduce.reduce.memory.mb to 8192

hdfs Safemode leave command

```
hdfs dfsadmin -safemode leave
```




43. Exercises

43.1 Docker Swarm Tutorial

Exercise 43.1 Develop a tutorial that deploys a Docker Swarm cluster on a number of ubuntu machines. Note that this may actually be easier as docker and docker swarm are distributed with recent versions of ubuntu. Just in case we are providing a link to an effort we found to install docker swarm. However we have not checked it or identified if it is useful.

- <https://rominirani.com/docker-swarm-tutorial-b67470cf8872>

43.2 Docker Swarm on Google Compute Engine

Exercise 43.2 Develop a tutorial that deploys a Docker Swarm cluster on Google Compute Engine. Note that this may actually be easier as docker and docker swarm are distributed with recent versions of ubuntu. Just in case we are providing a link to an effort we found to install docker swarm. However we have not checked it or identified if it is useful.

- <https://rominirani.com/docker-swarm-on-google-compute-engine-364765b400ed>

43.3 Single Node Hadoop

Exercise 43.3 At

<https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html>

you will find a comprehensive installation instruction that sets up a hadoop cluster on a single node.

- (a) Create a Dockerfile that deploys hadoop in a container
- (b) Develop sample applications and tests to test your cluster. You can use wordcount or similar.

43.4 Single Node Hadoop

Exercise 43.4 At

<https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-common/ClusterSetup.html>

you will find a comprehensive installation instruction that sets up a hadoop cluster in a distributed environment, can you use this set of instructions or identify other resources on the internet that allow the creation of a hadoop cluster on kubernetes. You can use docker compose for this.

- (a) Create docker compose and Dockerfiles that deploys hadoop in kubernetes
- (b) Develop sample applications and tests to test your cluster. You can use wordcount or similar.

43.5 Spark Cluster

Exercise 43.5 Develop a high quality tutorial that installes a spark cluster in kubernetes. Test your deployment on minikube but also Futuresystems echo.

You may want to get inspired from the talk *Scalable Spark Deployment using Kubernetes*:

- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-1/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-2/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-3/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-4/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-5/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-6/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-7/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-8/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-9/>

Make sure you do not plagiarize.



44. Docker Ecosystem

44.1 Docker Hub

status: 100

Docker Hub “is a cloud-based registry service” which provides a “centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline” [169]. There are both private and public repositories. Private repository can only be used by people within their own organization.

Docker Hub is “hardcoded into Docker as the default registry”, which means that the docker pull command will initialize the download automatically from Docker Hub [96]. It allows users to download (pull), build, test and store their images for easy deployment on any host they may have [169].

44.1.1 Create Docker ID and Log In

Log-in is not necessary for pulling Docker images from the Hub but necessary for push images. However when you want to store images on Docker hub you need to create an account. To create an account on Docker Hub, please visit the [Docker Hub main page](#). The free service will give you only one private Docker Hub Repository. In case you need more, you will need to upgrade to a paid plan.

For the rest of the tutorial we assume that you use the environment variable DUSER to indicate you username. It is easiest if you set it in your shell with

```
1 export DUSER=<PUT YOUR DOCKER USERNAME HERE>
```

44.1.2 Searching for Docker Images

There are two ways to search for Docker images on Docker Hub:

One way is to use the Docker command line tool. You could open an terminal and run the *docker search* command. For example, the following command searches for centOS images:

```
1 docker search centos
```

you will see output similar to:

```
1 | NAME                | DESCRIPTION                | STAR | OFFICIAL | AUTOMATED |
2 | -----|-----|-----|-----|-----|
3 | centos              | Official CentOS          | 4130 | [OK]     |           |
4 | ansible/centos7    | Ansible on Centos7      | 105  |          | [OK]     |
5 | ...
```

Official repositories are public, certified repositories from vendors and contributors to Docker. They contain Docker images from vendors like Canonical, Oracle, and Red Hat that you can use as the basis to build your applications and services. There is one official repository in this list, the first one, *centos*.

The other way is to search via the *Web Search Box* at the top of the Docker web page by typing the keyword. The search results can be sorted by number of stars, number of pulls, and whether it is an official image. Then for each search result, you can verify the information of the image by clicking the *details* button to make sure this is the right image that fits your needs.

44.1.3 Pulling Images

A particular image (take centos as an example) could be pulled using the following command:

```
1 docker pull centos
```

Tags could be used to specify the image to pull. By default the tag is latest, therefore the previous command is the same as the following:

```
1 docker pull centos:latest
```

You could use a different tag:

```
1 docker pull centos:6
```

To check the existing local docker images, run the following command:

```
1 docker images
```

The results show:

```
1 | REPOSITORY | TAG    | IMAGE ID    | CREATED      | SIZE  |
2 | -----|-----|-----|-----|-----|
3 | centos     | latest | 26cb1244b171 | 2 weeks ago | 195MB |
4 | centos     | 6      | 2d194b392dd1 | 2 weeks ago | 195MB |
```

44.1.4 Create Repositories

In order to push images to Docker Hub, you need to have a repository created.

When you first create a Docker Hub user, you see a *Get started with Docker Hub* screen, from which you can click directly into *Create Repository*. You can also use the *Create* menu to *Create Repository*. When creating a new repository, you can choose to put it in your Docker ID namespace, or that of any organization that you are in the owners team [170].

As an example, we created a repository `cloudtechnology` with the name space `$DUSER`. Hence the full name is `$DUSER/cloudtechnology`

44.1.5 Pushing Images

To push an image to the repository created, the following steps could be followed:

- Log into Docker Hub from the command line by specifying the username

```
1 docker login --username=$DUSER
```

Enter the password when prompted. If everything worked you will get a message similar to:

```
1 Login Succeeded
```

- Check image ID using:

```
1 docker images
```

the result looks similar to:

	REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
1					
2	-----	-----	-----	-----	-----
3	cloudmesh-nlp	latest	1f26a5f7a1b4	10 days ago	1.79GB
4	centos	latest	26cb1244b171	2 weeks ago	195MB
5	centos	latest	2d194b392dd1	2 weeks ago	195MB

and the image with ID `1f26a5f7a1b4` is the one to push to Docker Hub.

- Tag the image

```
1 docker tag 1f26a5f7a1b4 $DUSER/cloudmesh:firsttry
```

In general, a good choice for a tag is something that will help you understand what this container should be used in conjunction with, or what it represents.

- Now the list of images will look something like

	REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
1					
2	-----	-----	-----	-----	-----
3	cloudmesh-nlp	latest	1f26a5f7a1b4	10 d ago	1.79GB
4	\$DUSER/cloudmesh	firsttry	1f26a5f7a1b4	10 d ago	1.79GB
5	centos	latest	26cb1244b171	2 w ago	195MB
6	centos	latest	2d194b392dd1	2 w ago	195MB

- Push the image to the repository

```
1 docker push $DUSER/cloudmesh
```

It shows something similar to:

```
1 The push refers to repository [docker.io/$DUSER/cloudmesh]
2 18f9479cfc2c: Pushed
3 e9ddee98220b: Pushed
4 1d3522002590: Pushed
5 e3ab85ae555e: Pushed
6 bae105d9c555: Pushed
7 6b0c2fb2fe92: Pushed
8 c33cd8954775: Pushed
```

```
9 ecafeebb22db: Pushed
10 e0dbd107774a: Pushed
11 8cb07daea6f6: Pushed
12 db584c622b50: Mounted from library/ubuntu
13 52a7ea2bb533: Mounted from library/ubuntu
14 52f389ea437e: Mounted from library/ubuntu
15 88888b9b1b5b: Mounted from library/ubuntu
16 a94e0d5a7c40: Mounted from library/ubuntu
17 firsttry: digest: sha256:305b0f911077d9d6aab4b447b... size: 3463
```

Now the image is available on Docker Hub and everyone can pull it since it is a public repository by using command:

```
1 docker pull $DUSER/cloudmesh
```

44.1.6 Resources

- The official [Overview of Docker Hub](#) [169]
- Information about using docker repositories can be found at [Repositories on Docker Hub](#) [170]
- [How to Use DockerHub](#) [96]
- [Docker Tutorial Series](#)[333]

XIII

Cloud Virtual Machine Technologies

45	Introduction to Virtual Machines	505
45.1	QEMU and KVM	
45.2	Chameleon OpenStack	

TODO: TAs: Move all section includes here for releasable container lectures



45. Introduction to Virtual Machines

45.1 QEMU and KVM

We discuss the underlying virtualization technologies used by OpenStack and public cloud providers, such as AWS, Azure, and Google Cloud.

QEMU KVM (50)

45.2 Chameleon OpenStack

OpenStack on Chameleon delivers KVM based compute resources to provision virtual machines. It provides various image types on which we can deploy tools and software needed for the class and projects. We will you through the basic steps of getting access to OpenStack Chameleon cloud under the class allocation. Next, we will introduce you to the command line tools which you can use in your projects. Naturally using the GUI for your projects is not sufficient as setting up your environment will need steps to be executed by hand which is not sufficient. It is a goal of this class that you create your environment ins a reproducible fashion via scripts. Hence, although the Web interface called OpenStack Horizon is initially attractive, we should make sure to move on to the commandline interfaces. Furthermore, it is often difficult to resolve technical issues as the command line tools generate full debugging messages in case of issues and copy and past into helo windows is much easier and efficient than copy and past incomplete screenshots.

45.2.1 Outages

Any computer system may undergo maintenance. Before filing tickets with Chameleon cloud, make sure that the cloud is operational. Outages are posted at

<https://www.chameleoncloud.org/user/outages/>

To be notified by mail, you can subscribe to them at

<https://www.chameleoncloud.org/user/profile/subscriptions/>

45.2.2 Account Creation

The fist step to get access Chameleon cloud is to create a user account if you do not already have one. You can skip to the next section if you have a chameleon cloud account.

The register web page is available at:

<https://www.chameleoncloud.org/user/register/>

For more details, please als consult the chameleon chapter in the handbook.

45.2.3 Join a Project

An active project is required to access compute resources. Each class has a particular project number that you will need to write down as you will use it to interact with the system. The information is given out by the instructor.

For the Spring-18 classes including i524, e516, and e616 please use the following project number:

```
1 CH-819337
```

However, before you can access it the instructor (in our class Dr. von Laszewski) needs to authorize you to use the project. For this you have filled out an account survey in piazza. The most common errors we see are that students provide us with the wrong user name or have not applied for a chameleon account. Once the instructor has added you, you will be able to use VM's on Chameleon cloud.

45.2.4 Usage Restriction

As using VM's in a shared environment cost resources, you are **REQUIRED** to shut down your resources after you are not using them anymore. Furthermore, before the class is over and we assign grades you must terminate your instances and free all ip addresses. Remember that any running VM is just like you were running a real computer. I am sure you close the lid of your laptop when not in use. Shutting down the VM is similar and avoids that you unnecessarily use resources that others could use in a shared environment.

45.2.5 OpenStack RC File

We will use the Nova command line tools for Chameleon OpenStack and to authorize our account on the command line tools. To do so, you will need an openstack RC file.

Creating OpenStack RC via the editor

The easiest way is to create this file by hand while copying the following lines into the file `~/cloudmesh/chameleon/cc-openrc.sh`. Make sure that you place the file in a location you easily be found:

```
1 mkdir -p ~/cloudmesh/chameleon
```

The easiest way is to download a template from pur book with

```
1 wget https://raw.githubusercontent.com/cloudmesh/book/master/examples/↵
   ↵ chameleon/cc-openrc.sh -O ~/cloudmesh/chameleon/cc-openrc.sh
```

The `cc-openrc.sh` looks as follows:

```
1 #!/bin/bash
2
3 export CC_PROJECTID="CH-819337"
4 export CC_PREFIX="albert-111" # repalce with your username and hid number
5
6 export OS_AUTH_URL=https://openstack.tacc.chameleoncloud.org:5000/v2.0
7 # With Keystone you pass the keystone password.
8 echo "Please enter your OpenStack Password: "
9 read -sr OS_PASSWORD_INPUT
10 export OS_PASSWORD=$OS_PASSWORD_INPUT
11
12 export OS_TENANT_ID=$CC_PROJECTID
13 export OS_TENANT_NAME=$CC_PROJECTID
14 export OS_PROJECT_NAME=$CC_PROJECTID
15 export OS_USERNAME="<put your chameleon cloud username here>"
16
17
18 export OS_REGION_NAME="RegionOne"
19 if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
```

Please make sur to replace `<put your chameleon cloud username here>` with your chameleon cloud username. Now whenever you need top access chameleon cloud you can use the command

```
1 source ~/cloudmesh/chameleon/cc-openrc.sh
```

To simplify the configuration and documentation, we have included two shell environment variables. The first one is `CC_PROJECT`, that specifies the project number. The second one is a prefix that you

will use for VMS and keys as we are using a shared project. This way we can see which VMS and which keys have been uploaded and keep the names of them unique.

```
1 export CC_PROJECT=CH-819337
2 export CC_PREFIX=albert-111
```

Creating OpenStack RC via the GUI

In case you do not want to use the commandline option to obtain an RC sample, you can obtain the OpenStack RC file with the OpenStack Dashboard.

<https://openstack.tacc.chameleoncloud.org/dashboard>

Login and chose your project number for this project. Confirm your project number and find **Access & Security** on the left menu. The Access & Security page has tabs and choose **API Access** to download credentials on a local machine. Click **Download OpenStack RC File** to download *CH-\$PROJECTID-openrc.sh* file on your machine (see Figure 45.1). Every time you use nova command line tools, the file should be loaded on your terminal.

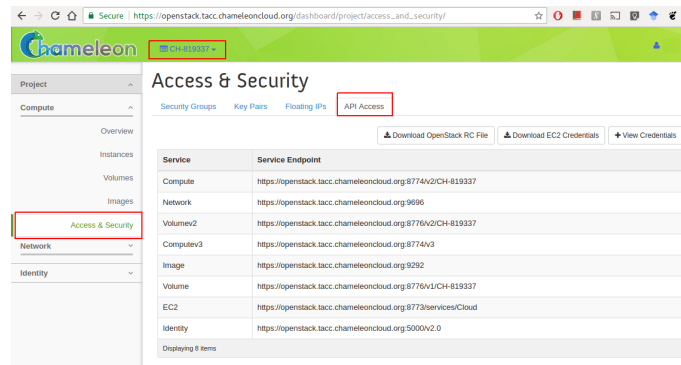


Figure 45.1: Access and Security GUI

```
1 mkdir -p ~/.cloudmesh/chameleon
2 mv ~/Downloads/CH-$CC_PROJECT-openrc.sh ~/.cloudmesh/chameleon/cc-openrc.sh
```

Just as in the previous section please add the following to your *openrc.sh* file while adapting it appropriately.

```
1 export CC_PROJECT=CH-819337
2 export CC_PREFIX=albert-111
```

Once you *source* the file, you can use nova command line tools without sourcing it again. The environment variables are enabled while your terminal is alive. In case you have not stored the original RC file in the Downloads folder, please copy it from that location instead.

45.2.6 CLI to Manage Virtual Machines

OpenStack provides a commandline tool called *nova* to manage virtual machines. To install it please use the command

```
1 pip install python-openstackclient
```

To see if your configuration works and the command is installed, make sure you have the *cc-openrc.sh* file and sourced it. Than issue the command

```
1 $ nova image-list
```

You will see an output similar to

```
1 +-----+-----+-----+-----+
2 | ID                | Name                | Status | Server |
3 +-----+-----+-----+-----+
4 | be46bd5a-c4a5-4495-ad30-35618... | CC-C7-autologin    | ACTIVE |        |
5 | 1fe5138b-300b-4b30-8d22-e7287... | CC-CentOS7         | ACTIVE |        |
6 ...
```

45.2.7 Creating SSH keys

Naturally you will need an ssh key. If you do not have an existing SSH keypair, you can create one. Please see Section 16 for more details:

```
1 ssh-keygen -t rsa -C albert@example.edu
```

45.2.8 KeyPair Registration

Once you have completed the installation of nova, you also need to register a ssh keypair with openstack to be able to log into the virtual machines that you start. To register your public key, use:

```
1 nova keypair-add --pub-key ~/.ssh/id_rsa.pub $CC_PREFIX-key
```

Once you register your key, you can confirm if your key registration has been successful by listing the keys:

```
1 nova keypair-list
```

You will see an output similar to:

```
1 +-----+-----+-----+
2 | Name                | Fingerprint          |
3 +-----+-----+-----+
4 | $CC_PREFIX-key     | cf:04:06:aa:8b:76:af:77:aa:0a:b5:87:ff:0f:ba:97 |
5 +-----+-----+-----+
```

45.2.9 Start a new VM instance

To start new instances you can use the *nova boot* command. It will start a VM instance. You can use some parameters to specify which base image and a server size we will use with a name. We use *CC-Ubuntu16.04* base image in this tutorial which is an official Ubuntu 16.04 image provided by Chameleon project.

```
1 nova boot --image CC-Ubuntu16.04 --key-name $CC_PREFIX-key --flavor m1.↵
↵ small $CC_PREFIX-01
```

where the 01 indicates the instance number. Note that we will be terminating and deleting any VM in our project that does not follow this naming convention.

45.2.10 Floating IP Address

If your new VM instance is up and running, it needs an external ip address which is also called floating IP address. A floating IP allows you to get access to this VM from the internet. Note that

chameleon has a limited number of floating IP addresses and it is best to return them if not in use. If chameleon runs out of floating IP addresses, please submit a ticket to chameleon. However in many cases the VM may only need an internal IP address as a default. In case you need to access others, you could even tunnel all connections through a single floating IP. naturally this would limit data transfers in and out of chameleon, but is a recommended way to deal with limited floating IPs.

Let us showcase how to associate a floating IP address and access it via SSH.

```

1 nova floating-ip-create ext-net
2 +-----+-----+-----+-----+
3 | Id           | IP           | Server Id | Fixed IP | Pool      |
4 +-----+-----+-----+-----+
5 | 13dc309e-... | 129.114.111.37 | -         | -         | ext-net   |
6 +-----+-----+-----+-----+

```

Now we have a IP address to assign to a VM instance. In this tutorial, we will associate *129.114.111.37* to our *albert-111-01* VM instance by:

```

1 nova floating-ip-associate albert-111-01 129.114.111.37

```

Once you completed this step, you are now able to SSH into your VM instance. Confirm *ACTIVE* state in your VM to get access.

```

1 | f19e1... | albert-111-01 | ACTIVE | - | Running | $CC_PROJECT-net= |
2 |          |                |        |   |          | 192.168.0.13, |
3 |          |                |        |   |          | 129.114.111.37 |

```

where 111 is the number from your hid and 01 is the instance number

```

1 ssh cc@129.114.111.37

```

Note that *cc* is login name your VM if you start a VM with the official Chameleon cloud image.

45.2.11 Termination of VM Instance

If you completed your work on your VM instance, you have to terminate your VM and release a floating IP address associated with. For example, we terminate our first instance and the IP address by:

```

1 nova delete $CC_PREFIX-01
2 nova floating-ip-delete 129.114.111.37

```

Please note that when using *delete* you will delete the VM. In case you still need to use it use *stop* and to restart it use *start* instead.



MapReduce

46	Hadoop	513
46.1	Hadoop Motivation	
46.2	Hadoop and MapReduce	
46.3	Hadoop EcoSystem	
46.4	Hadoop Components	
46.5	Hadoop and the Yarn Resource Manager	
46.6	PageRank	
46.7	Installation of Hadoop	
47	Spark	519
47.1	Motivation for Spark	
47.2	Spark RDD Operations	
47.3	Spark DAG	
47.4	Spark vs. other Frameworks	
47.5	Installation of Spark	
47.6	Spark Streaming	
48	Draft: Harp	537
48.1	Harp	
49	Draft: Twister	539
49.1	Twister2 Examples	
49.2	Twister2 Installation	



46. Hadoop

Hadoop is an open source framework for storage and processing of large datasets on commodity clusters. Hadoop internally uses its own file system called HDFS (Hadoop Distributed File System).

46.1 Hadoop Motivation

In this section we discuss about the background of Mapreduce along with Hadoop and core components of Hadoop.


TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Hadoop A (19:02) 

46.2 Hadoop and MapReduce

In this section we discuss about the usage Hadoop MapReduce architecture.


TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Hadoop B (13:19) 

46.3 Hadoop EcoSystem

In this section we discuss about the Hadoop EcoSystem and the architecture.

TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Hadoop C (12:57) 

46.4 Hadoop Components

In this section we discuss about Hadoop Components in detail.

TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Hadoop D (15:14) 

46.5 Hadoop and the Yarn Resource Manager

In this section we discuss about Yarn resource manager and novel components added to the Hadoop framework in case of improving the performance and minimizing fault tolerance.

TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Hadoop E (14:55) 

46.6 PageRank

In this section we discuss about a real world problem that can be solved using the MapReduce technique. PageRank is a problem solved by the earliest stages of the Google.inc. In this section we discuss about the theoretical background about this problem and we discuss how this can be solved using the map reduce concepts.

TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Hadoop E (15:14) 

46.7 Installation of Hadoop

In this section we use Hadoop 3.0.1 and we install Hadoop locally in Ubuntu 16.04. We also describe the installation of the Yarn resource manager. We assume that you have ssh, and rsync installed and use emacs as editor.

46.7.1 Prerequisites

```
1 sudo apt-get install ssh
2 sudo apt-get install rsync
3 sudo apt-get install emacs
```

46.7.2 User and User Group Creation

For security reasons we will install hadoop in a particular user and user group. We will use the following

```
1 sudo addgroup hadoop_group
2 sudo adduser --ingroup hadoop_group hduser
3 sudo adduser hduser sudo
```

These steps will provide sudo privileges to the created hduser user and add the user to the group hadoop_group.

46.7.3 Configuring SSH

Note

Here we configure SSH key for the local user to install hadoop with a ssh-key. This is different from the ssh-key you used for Github, FutureSystems, etc. Follow this section to configure it for Hadoop installation.

The ssh content is included here because, we are making a ssh key for this specific user. Next, we have to configure ssh to be used by the hadoop user.

```
1 su - hduser

1 ssh-keygen -t rsa
```

Follow the instructions as provided in the commandline. When you see the following console input, press ENTER. Here only we will create password less keys. IN general this is not a good idea, but for this case we make an exception.

```
1 Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
```

Next you will be asked to enter a password for ssh configuration,

```
1 Enter passphrase (empty for no passphrase):
```

Here enter the same password

```
1 Enter same passphrase again:
```

Finally you will see something like this after these steps are finished.

```

1 Generating public/private rsa key pair.
2 Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
3 Created directory '/home/hduser/.ssh'.
4 Enter passphrase (empty for no passphrase):
5 Enter same passphrase again:
6 Your identification has been saved in /home/hduser/.ssh/id_rsa.
7 Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
8 The key fingerprint is:
9 SHA256:0UBCPd6oYp7MEzCp0hMhNiJyQo6PaPCDu0T48xUDDc0 hduser@computer
10 The key's randomart image is:
11 +---[RSA 2048]----+
12 |      .+ooo      |
13 | .    oE.o      |
14 |+  .. ...+.    |
15 |X+=   . o..    |
16 |XX.o  o.S      |
17 |Bo+ + .o      |
18 |*o * +.      |
19 |*.. *        |
20 | +.o..      |
21 +-----[SHA256]-----+

```

You have successfully configured ssh.

46.7.4 Installation of Java

If you are already logged into su, you can skip the next command:

```
1 su - hduser
```

Now execute the following commands to download and install java

```

1 mkdir -p ~/cloudmesh/bin
2 cd ~/cloudmesh/bin
3 wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" "http↵
↵ ://download.oracle.com/otn-pub/java/jdk/8u161-b12/2↵
↵ f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz"
4 tar xzvf jdk-8u161-linux-x64.tar.gz

```

46.7.5 Installation of Hadoop

First we will take a look on how to install Hadoop 3.0.1 on Ubuntu 16.04. We may need a prior folder structure to do the installation properly.

```

1 cd ~/cloudmesh/bin/
2 wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-3.0.1/hadoop↵
↵ -3.0.1.tar.gz
3 tar -xzvf hadoop-3.0.1.tar.gz

```

46.7.6 Hadoop Environment Variables

In Ubuntu the environmental variables are setup in a file called bashrc at it can be accessed the following way

```
1 emacs ~/.bashrc
```

```

1 export JAVA_HOME=~/.cloudmesh/bin/jdk1.8.0_161
2 export HADOOP_HOME=~/.cloudmesh/bin/hadoop-3.0.1
3 export YARN_HOME=$HADOOP_HOME
4 export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
5 export PATH=$HADOOP_HOME/bin:$JAVA_HOME/bin:$PATH

```

In Emacs to save the file Ctrl-X-S and Ctrl-X-C to exit. After editing you must update the variables in the system.

```

1 source ~/.bashrc
2 java -version

```

If you have installed things properly there will be no errors. It will show the version as follows,

```

1 java version "1.8.0_161"
2 Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
3 Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)

```

And verifying the hadoop installation,

```

1 hadoop

```

If you have successfully installed this, there must be a message shown as below.

```

1 Usage: hadoop [--config confdir] COMMAND
2     where COMMAND is one of:
3     fs                run a generic filesystem user client
4     version           print the version
5     jar <jar>        run a jar file
6     checknative [-a|-h] check native hadoop and compression libraries ↔
7     ↪ availability
8     distcp <srcurl> <desturl> copy file or directories recursively
9     archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop ↔
10    ↪ archive
11    classpath         prints the class path needed to get the
12    credential        interact with credential providers
13    daemonlog        Hadoop jar and the required libraries
14    trace             get/set the log level for each daemon
15    or                view and modify Hadoop tracing settings
16    CLASSNAME        run the class named CLASSNAME
17 Most commands print help when invoked w/o parameters.

```





47. Spark

This section covers an introduction to Spark that is split up into eight parts. We discuss Spark background, RDD operations, Shark, Spark ML, Spark vs Other Frameworks.

47.1 Motivation for Spark

In this section we discuss about the background of Spark and core components of Spark.


TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

[Spark A \(15:57\)](#) 

47.2 Spark RDD Operations

In this section we discuss about the background of RDD operations along with other transformation functionalities in Spark.


TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

[Spark B \(12:17\)](#) 

47.3 Spark DAG

In this section we discuss about the background of DAG (direct acyclic graphs) operations along with other components like Shark in the earlier stages of Spark.


TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Spark C (10:37) 

47.4 Spark vs. other Frameworks

In this section we discuss about the real world applications that can be done using Spark. And also we discuss some comparison results obtained from experiments done in Spark along with Frameworks like Harp, Harp DAAL, etc. We discuss the benchmarks and performance obtained from such experiments.

TODO: Gregor: Update Video Hadoop Urls to Youtube Urls

Spark D (26:18) 

47.5 Installation of Spark

In this section we will discuss how to install Spark 2.3.0 in Ubuntu 16.04.

47.5.1 Prerequisites

We assume that you have ssh, and rsync installed and use emacs as editor.

```
1 sudo apt-get install ssh
2 sudo apt-get install rsync
3 sudo apt-get install emacs
```

47.5.2 Installation of Java

First download Java 8.

```
1 mkdir -p ~/cloudmesh/bin
2 cd ~/cloudmesh/bin
3 wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" "http↔
↔ ://download.oracle.com/otn-pub/java/jdk/8u161-b12/2↔
↔ f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz"
4 tar xvzf jdk-8u161-linux-x64.tar.gz
```

Then add the environmental variables to the bashrc file.

```
1 emacs ~/.bashrc

1 export JAVA_HOME=~/cloudmesh/bin/jdk1.8.0_161
2 export PATH=$JAVA_HOME/bin:$PATH
```

Source the bashrc file after adding the environmental variables.

```
1 source ~/.bashrc
```

47.5.3 Install Spark with Hadoop

Note

Here we use Spark packaged with Hadoop. In this package Spark uses Hadoop 2.7.0 in the packaged version. Note that in Section 46.7 we use for the vanilla Hadoop installation Hadoop 3.0.0.

Create the base directories and go to the directory.

```
1 mkdir -p ~/cloudmesh/bin
2 cd ~/cloudmesh/bin
```

Then download Spark 2.3.0 as follows.

```
1 wget https://www.apache.org/dyn/closer.lua/spark/spark-2.3.0/spark-2.3.0-↔
↔ bin-hadoop2.7.tgz
```

Now extract the file,

```
1 tar xzf spark-2.3.0-bin-hadoop2.7.tgz
2 mv spark-2.3.0-bin-hadoop2.7 spark-2.3.0
```


Resource	Source
/home/vibhatha/cloudmesh/bin/hadoop-3.0.0/etc/hadoop/	System Classpath
/home/vibhatha/cloudmesh/bin/hadoop-3.0.0/share/hadoop/common/hadoop-common-3.0.0-tests.jar	System Classpath
/home/vibhatha/cloudmesh/bin/hadoop-3.0.0/share/hadoop/common/hadoop-common-3.0.0.jar	System Classpath

Figure 47.1: Spark Web UI - Hadoop Path

```
1 mkdir -p ~/cloudmesh/bin
2 cd ~/cloudmesh/bin
```

Then download Spark 2.3.0 as follows.

```
1 wget http://mirrors.ibiblio.org/apache/spark/spark-2.3.0/spark-2.3.0-bin-↵
   ↵ without-hadoop.tgz
```

Now extract the file,

```
1 tar xzf spark-2.3.0-bin-without-hadoop.tgz
```

Then add the environmental variables,

Note

If you have already installed Spark with Hadoop by following section [47.5.3](#) please update the current SPARK HOME variable with the new path.

```
1 emacs ~/.bashrc
```

Go to the last line and add the following content.

```
1 export SPARK_HOME=~/cloudmesh/bin/spark-2.3.0-bin-without-hadoop
2 export PATH=$SPARK_HOME/bin:$PATH
```

Source the bashrc file.

```
1 source ~/.bashrc
```

47.5.7 Configuring Hadoop

Now we must add the current Hadoop version that we are using for Spark. Open up a new terminal and then run the following.

```
1 cd $SPARK_HOME

1 cd conf
2 cp spark-env.sh.template spark-envn.sh
```

Now we need to add a new line to show the current path to hadoop installation. Add the following variable in to the spark-env.sh file.

```
1 emacs spark-env.sh
2 export SPARK_DIST_CLASSPATH=$(($HADOOP_HOME/bin/hadoop classpath)
```

47.5.8 Test Spark Installation

Open up a new terminal and then run the following command.

47.6 Spark Streaming

47.6.1 Streaming Concepts

Spark Streaming is one of the components extending from Core Spark. Spark streaming provides a scalable fault tolerant system with high throughput. For streaming data into spark, there are many libraries like Kafka, Flume, Kinesis, etc.

47.6.2 Simple Streaming Example

In this section, we are going to focus on making a simple streaming application using the network in your computer. Here we are going to expose a particular port and from that port we are going to continuously stream data by user entries and the word count is being calculated as the output.

First, create a Makefile

```
1 mkdir -p ~/cloudmesh/spark/examples/streaming
2 cd ~/cloudmesh/spark/examples/streaming
3 emacs Makefile
```

Then add the following content to Makefile.

Note

Please add a tab when adding the corresponding command for a given instruction in Makefile. In pdf mode the tab is not clearly shown.

```
1 SPARKHOME = ${SPARK_HOME}
2 run-streaming:
3   ${SPARKHOME}/bin/spark-submit streaming.py localhost 9999
```

Now we need to create file called streaming.py

```
1 emacs streaming.py
```

Then add the following content.

```
1 from pyspark import SparkContext
2 from pyspark.streaming import StreamingContext
3
4 # Create a local StreamingContext with two working thread and batch ←
   ↪ interval of 1 second
5 sc = SparkContext("local[2]", "NetworkWordCount")
6
7 log4jLogger = sc._jvm.org.apache.log4j
8 LOGGER = log4jLogger.LogManager.getLogger(__name__)
9 LOGGER.info("Pyspark script logger initialized")
10
11 ssc = StreamingContext(sc, 1)
12
13
14 # Create a DStream that will connect to hostname:port, like localhost:9999
15 lines = ssc.socketTextStream("localhost", 9999)
16 # Split each line into words
17 words = lines.flatMap(lambda line: line.split(" "))
```

```

18 # Count each word in each batch
19 pairs = words.map(lambda word: (word, 1))
20 wordCounts = pairs.reduceByKey(lambda x, y: x + y)
21
22 # Print the first ten elements of each RDD generated in this DStream to the←→
    ↪ console
23 wordCounts.pprint()
24 ssc.start() # Start the computation
25 ssc.awaitTermination(100) # Wait for the computation to terminate

```

To run the code, we need to open up two terminals.

Terminal 1 :

First use netstat to open up a port to start communication.

```
1 nc -lk 9999
```

Terminal 2 :

Now run the Spark programme in the second terminal.

```
1 make run-streaming
```

In this terminal you can see a script running trying to read the stream coming from the port 9999. You can enter texts in the Terminal 1 and these texts will be tokenized and the word count is calculated and the result is shown in the Terminal 2.

47.6.3 Spark Streaming For Twitter Data

In this section we are going to learn how to use Twitter data as the streaming data source and use Spark Streaming capabilities to process the data. As the first step you must install the python packages using pip.

Step 1

```
1 sudo pip install tweepy
```

Step 2

Then you need to create an account in Twitter Apps. Go to

- <https://apps.twitter.com/>

and sign in to your twitter account or create a new twitter account. Then you need to create a new application, let's name this application as Cloudmesh-Spark-Streaming.

First you need to create an app with the app name we suggested in this section. The way to create the app is mentioned in Figure 47.6.3.

Next we need to take a look at the dashboard created for the app. You can see how your dashboard looks like in Figure 47.6.3.

Next the application tokens generated must be reviewed and it can be found in Figure 47.6.3, you need to go to the Keys and Access Tokens tab.

Now you need to generate the access tokens for the first time if you have not generated access tokens and this can be done by clicking the Create my access token button.

Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? [OAuth 1.0a](#) applications should explicitly specify their `oauth_callback` URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement


Yes, I have read and agree to the [Twitter Developer Agreement](#).

Figure 47.2: Create Twitter App

Your application has been created. Please take a moment to review and adjust your application's settings.

Cloudmesh-Spark-Streaming

Details
Settings
Keys and Access Tokens
Permissions



Testing Spark Streaming

<https://www.vibhatha.org>

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Figure 47.3: Go To Twitter App Dashboard

Cloudmesh-Spark-Streaming

[Test OAuth](#)[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	<input type="text"/>
Consumer Secret (API Secret)	<input type="text"/>
Access Level	Read and write (modify app permissions)
Owner	<input type="text"/>
Owner ID	<input type="text"/>

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

[Create my access token](#)

Figure 47.4: Create Your Twitter Settings

Cloudmesh-Spark-Streaming

[Test OAuth](#)[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	<input type="text"/>
Consumer Secret (API Secret)	<input type="text"/>
Access Level	Read and write (modify app permissions)
Owner	<input type="text"/>
Owner ID	<input type="text"/>

Application Actions

[Regenerate Consumer Key and Secret](#)[Change App Permissions](#)

Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

[Create my access token](#)

Figure 47.5: Create Your Twitter Access Tokens

Note

The access tokens and keys are blurred in this section for privacy issues.

Step 3

Let us build a simple Twitter App to see if everything is okay.

```
1 mkdir -p ~/cloudmesh/spark/streaming
2 cd ~/cloudmesh/spark/streaming
3 emacs twitterstreaming.py
```

Add the following content to the file and make sure you update the corresponding token keys with your token values.

```
1 import tweepy
2
3 CONSUMER_KEY = 'your_consumer_key'
4 CONSUMER_SECRET = 'your_consumer_secret'
5 ACCESS_TOKEN = 'your_access_token'
6 ACCESS_TOKEN_SECRET = 'your_access_token_secret'
7
8 auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
9 auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
10 api = tweepy.API(auth)
11
12 status = "Testing!"
13 api.update_status(status=status)
```

```
1 python twitterstreaming.py
```

Step 4

Let us start the twitter streaming exercise. We need to create a Tweet Listener in order to retrieve data from twitter regarding a topic of your choice. In this exercise, we have tried keywords like trump, indiana, messi.

```
1 mkdir -p ~/cloudmesh/spark/streaming
2 cd ~/cloudmesh/spark/streaming
3 emacs tweetlistener.py
```

Note

Make your to replace strings related to secret keys and ip addresses by replacing these values depending on your machine configuration and twitter keys.

Now add the following content.

```
1 import tweepy
2 from tweepy import OAuthHandler
3 from tweepy import Stream
4 from tweepy.streaming import StreamListener
5 import socket
6 import json
7
8 CONSUMER_KEY = 'YOUR_CONSUMER_KEY'
9 CONSUMER_SECRET = 'YOUR_CONSUMER_SECRET'
10 ACCESS_TOKEN = 'YOUR_ACCESS_TOKEN'
```

```

11 ACCESS_SECRET = 'YOUR_SECRET_ACCESS'
12
13 class TweetListener(StreamListener):
14
15     def __init__(self, csocket):
16         self.client_socket = csocket
17
18     def on_data(self, data):
19         try:
20             msg = json.loads( data )
21             print( msg['text'].encode('utf-8') )
22             self.client_socket.send( msg['text'].encode('utf-8') )
23             return True
24         except BaseException as e:
25             print("Error on_data: %s" % str(e))
26         return True
27
28     def on_error(self, status):
29         print(status)
30         return True
31
32 def sendData(c_socket):
33     auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
34     auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)
35
36     twitter_stream = Stream(auth, TweetListener(c_socket))
37     twitter_stream.filter(track=['messi']) # you can change this topic
38
39 if __name__ == "__main__":
40     s = socket.socket()
41     host = "YOUR_MACHINE_IP"
42     port = 5555
43     s.bind((host, port))
44
45     print("Listening on port: %s" % str(port))
46
47     s.listen(5)
48     c, addr = s.accept()
49
50     print( "Received request from: " + str( addr ) )
51
52     sendData( c )

```

step 5**Note**

Please replace the local file paths mentioned in this code with a file path of your preference depending on your workstation. And also IP address must be replaced with your ip address. The log folder path must be pre-created and makesure to replace the registerTempTable name with respect to the entity that you are referring. This will minimize the conflicts among different topics when you need to plot it in a simple manner.

Add the following content to the IpythonNote book as follows

Open up a terminal,

```

1 cd ~/cloudmesh/spark/streaming
2 jupyter notebook

```

Then in the browser the jupyter notebook is being loaded. There create a new Ipython notebook called twittersparkstremer.

Then add the following content.

```

1 from pyspark import SparkContext
2 from pyspark.streaming import StreamingContext
3 from pyspark.sql import SQLContext
4 from pyspark.sql.functions import desc

1 sc = SparkContext('local[2]', 'twittersparkstreamer')

1 ssc = StreamingContext(sc, 10 )
2 sqlContext = SQLContext(sc)
3 ssc.checkpoint( "file:///home/<your-username>/cloudmesh/spark/streaming/↔
↔ logs/messi")

1 socket_stream = ssc.socketTextStream("YOUR_IP_ADDRESS", 5555)

1 lines = socket_stream.window( 20 )

1 from collections import namedtuple
2 fields = ("tag", "count" )
3 Tweet = namedtuple( 'Tweet', fields )

1 ( lines.flatMap( lambda text: text.split( " " ) )
2 .filter( lambda word: word.lower().startswith("#") )
3 .map( lambda word: ( word.lower(), 1 ) )
4 .reduceByKey( lambda a, b: a + b )
5 .map( lambda rec: Tweet( rec[0], rec[1] ) )
6 .foreachRDD( lambda rdd: rdd.toDF().sort( desc("count") )
7 .limit(10).registerTempTable("tweetsmessi") ) )#change table ↔
↔ name depending on your entity

1 sqlContext

1 <pyspark.sql.context.SQLContext at 0x7f51922ba350>

1 ssc.start()

1 import matplotlib.pyplot as plt
2 import seaborn as sn

1 import time
2 from IPython import display
3
4
5 count = 0
6 while count < 10:
7     time.sleep( 20 )
8     top_10_tweets = sqlContext.sql( 'Select tag, count from tweetsmessi' ) #↔
↔ change table name according to your entity
9     top_10_df = top_10_tweets.toPandas()
10    display.clear_output(wait=True)

```

```

11 #sn.figure( figsize = ( 10, 8 ) )
12 sn.barplot( x="count", y="tag", data=top_10_df)
13 plt.show()
14 count = count + 1

1 ssc.stop()

```

step 6

Open Terminal 1, then do the following

```

1 cd ~/cloudmesh/spark/streaming
2 python tweetslistener.py

```

It will show that:

```

1 Listening on port: 5555

```

Open Terminal 2

Now we must start the Spark app by running the content in the IPython Notebook by pressing SHIFT-ENTER in each box to run each command. Make sure not to run twice the starting command of the SparkContext or initialization of SparkContext.

Now you will see streams in the Terminal 1 and you can see plots after a while in the IPython Notebook.

Sample outputs can be seen in the Figures [47.6.3](#), [47.6.3](#), [47.6.3](#), [47.6.3](#).

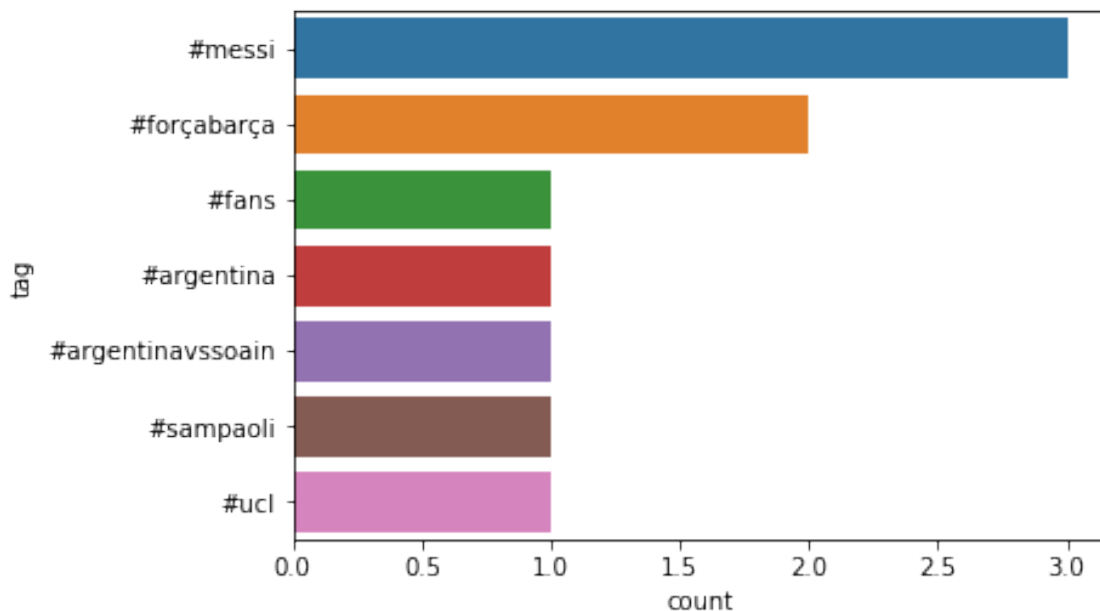


Figure 47.6: Twitter Topic Messi

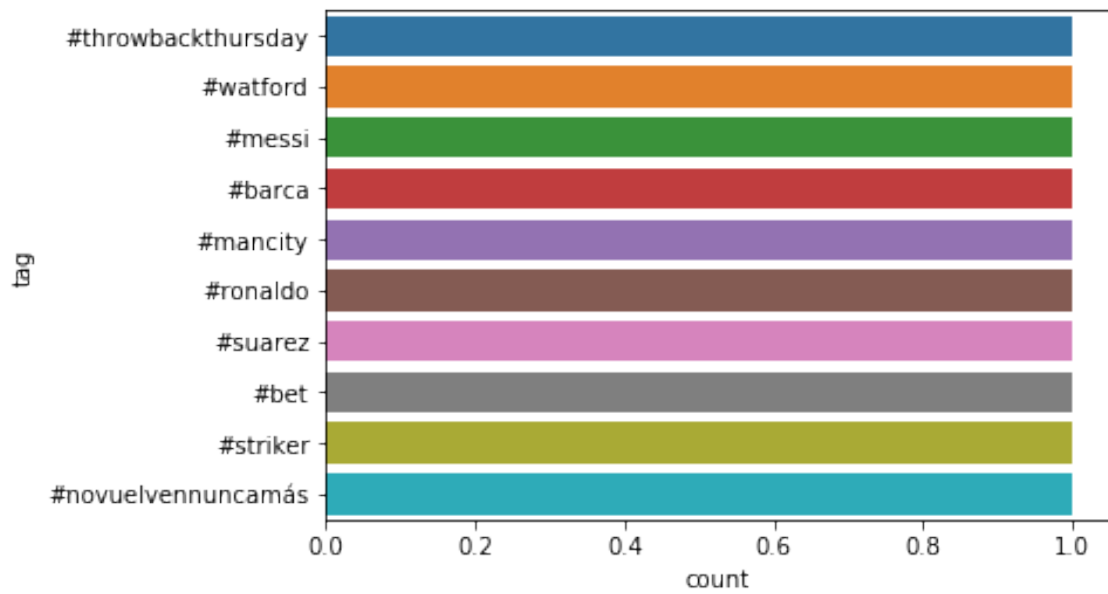


Figure 47.7: Twitter Topic Messi

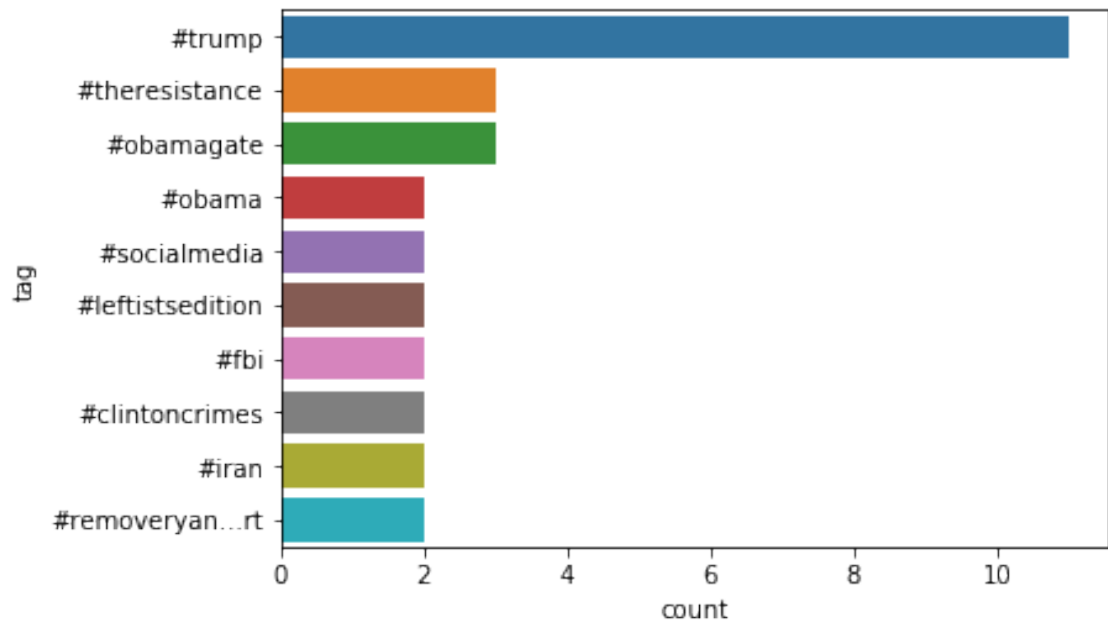


Figure 47.8: Twitter Topic Messi

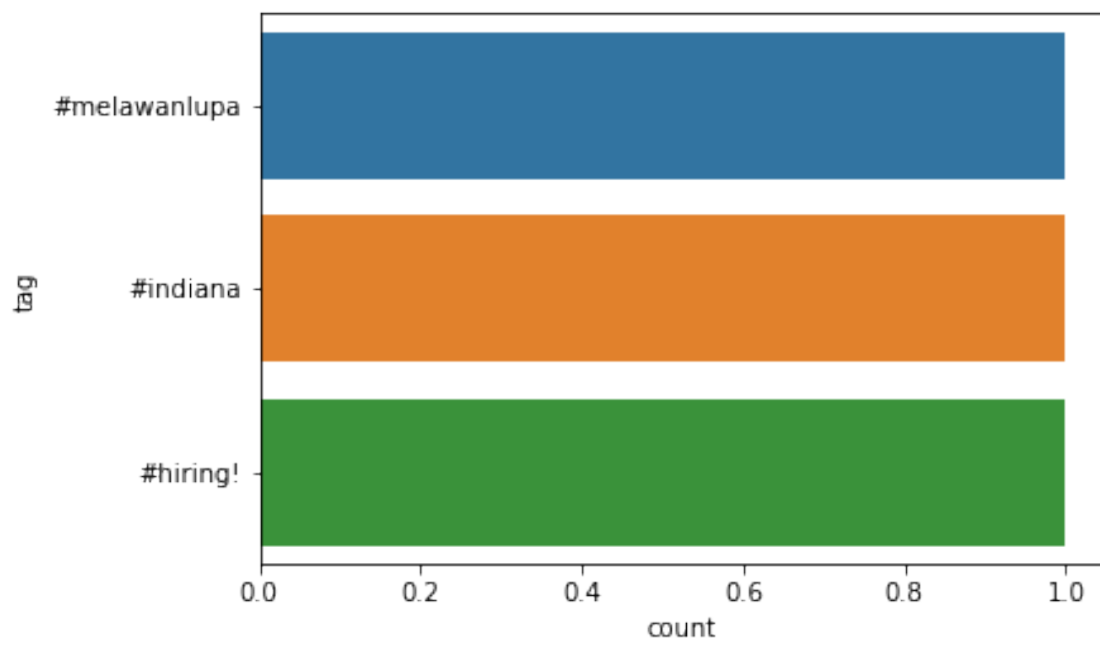


Figure 47.9: Twitter Topic Messi



48. Draft: Harp

48.1 Harp

TODO: TA: complete harp section



49. Draft: Twister

49.1 Twister2 Examples

TODO: TA: complete twister 2

In this section we will be discussing some examples from Twister2 Framework.

49.1.1 Introduction to Twister2

49.1.2 Twister2 Examples

49.2 Twister2 Installation

49.2.1 Prerequisites

- Operating Systems = [Red Hat Enterprise Linux Server release 7, Ubuntu 14.04 , Ubuntu 16.04] We only use Ubuntu 16.04
- Java (Jdk 1.8) Covered in Section 46.7.
- G++ Compiler `sudo apt-get install g++`
- Maven Installation Explained in Section 49.2.1
- OpenMPI Installation Explained in Section 49.2.1
- Bazel Build Installation Explained in Section 49.2.1
- Additional Libraries Explained in Section??

Maven Installation

Execute the following commands to install Maven locally.

```
1 mkdir -p ~/cloudmesh/bin/maven
2 cd ~/cloudmesh/bin/maven
3 wget http://mirrors.ibiblio.org/apache/maven/maven-3/3.5.2/binaries/↔
  ↪ apache-maven-3.5.2-bin.tar.gz
4 tar xzf apache-maven-3.5.2-bin.tar.gz
```

Adding environmental variables

```
1 emacs ~/.bashrc
```

Add the following line at the end of the file.

```
1 MAVEN_HOME=~/cloudmesh/bin/maven/apache-maven-3.5.2
2 PATH=$MAVEN_HOME/bin:$PATH
3 export MAVEN_HOME PATH
```

```
1 source ~/.bashrc
```

OpenMPI Installation

For Twister2, the recommended version is Open MPI 3.0.0. Execute the following commands to install Open MPI locally.

```
1 mkdir -p ~/cloudmesh/bin/openmpi
2 cd ~/cloudmesh/bin/openmpi
3 wget https://www.open-mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.↔
  ↪ tar.gz
```

Add environmental variables to bashrc file.

```
1 emacs ~/.bashrc
2 mkdir ~/cloudmesh/bin/openmpi/build
3 BUILD=~/cloudmesh/bin/openmpi/build
4 OMPI_300=~/cloudmesh/bin/openmpi
5 PATH=$BUILD/bin:$PATH
6 LD_LIBRARY_PATH=$BUILD/lib:$LD_LIBRARY_PATH
7 export BUILD OMPI_300 PATH LD_LIBRARY_PATH
```

```
1 source ~/.bashrc
```

Next build the OpenMPI 3.0.0,

```
1 cd $OMPI_300
2 ./configure --prefix=$BUILD --enable-mpi-java
3 make;make install
```

Make sure the installation is successful by executing,

```
1 mpirun --version
```

Then you will see an output,

```
1 mpirun (Open MPI) 3.0.0
```

Install the following command to add this as a maven artifact,

```
1 mvn install:install-file -DcreateChecksum=true -Dpackaging=jar -Dfile=↔
↔ $OMPI_300/mpi/mpi/java/java/mpi.jar -DgroupId=mpi -DartifactId=↔
↔ mpijavabinding -Dversion=3.0.0
```

Bazel Installation

For this installation, Bazel 0.8.1 is recommended. Execute the following commands to install Bazel,

```
1 mkdir -p ~/cloudmesh/bin/bazel
2 cd ~/cloudmesh/bin/bazel
3 wget https://github.com/bazelbuild/bazel/releases/download/0.8.1/bazel↔
↔ -0.8.1-installer-linux-x86_64.sh
4 chmod +x bazel-0.8.1-installer-linux-x86_64.sh
5 ./bazel-0.8.1-installer-linux-x86_64.sh --user
6 export PATH=$HOME/bin:$PATH
```

Install Extras

Install the other requirements as follows,

```
1 sudo apt-get install git build-essential automake cmake libtool-bin zip ↔
↔ libunwind-setjmp0-dev zlib1g-dev unzip pkg-config python-setuptools ↔
↔ -y
2 sudo apt-get install python-dev python-pip
```

Now you have successfully installed the required packages. Let us compile Twister2.

49.2.2 Compiling Twister2

First clone the repository from Github.

```
1 mkdir ~/cloudmesh/twister2
2 cd ~/cloudmesh/twister2
3 git clone git@github.com:DSC-SPIDAL/twister2.git
4 cd twister2
```

Now compile the code as follows,

```
1 bazel build --config=ubuntu twister2/...
```

In order to build packages run the following commands,

```
1 bazel build --config=ubuntu //scripts/package:tarpkgs
```

You can extract the `bazel-bin/scripts/package/twister2-client.tar.gz` to run Twister2.



Draft: Cloud Data Management

50	Data Formats	549
50.1	YAML	
50.2	JSON	
50.3	XML	
51	NoSQL	551
51.1	RDBMS vs. NoSQL	
51.2	NoSQL Characteristics	
51.3	BigTable	
51.4	HBase	
51.5	HBase Coding	
51.6	Draft: MongoDB	
51.7	Indexing Applications	
51.8	Related Work	
51.9	Indexexamples	
51.10	Indexing 101	
51.11	Social Media Searches	
51.12	Analysis Algorithms	



50. Data Formats

50.1 YAML

The term *YAML* stand for “YAML Ain’t Markup Language”. According to the Web Page at

- <http://yaml.org/>

“YAML is a human friendly data serialization standard for all programming languages.” There are multiple versions of YAML existing and one needs to take care of that your software supports the right version. The current version is YAML 1.2.

YAML is often used for configuration and in many cases can also be used as XML replacement. Important is that YAML in contrast to XML removes the tags while replacing them with indentation. This has naturally the advantage that it is more easily to read, however, the format is strict and needs to adhere to proper indentation. Thus it is important that you check your YAML files for correctness, either by writing for example a python program that read your yaml file, or an online YAML checker such as provided at

- <http://www.yamllint.com/>

An example on how to use yaml in python is provided in Figure 50.1. Please note that YAML is a superset of JSON. Originally YAML was designed as a markup language. However as it is not document oriented but data oriented it has been recast and it does no longer classify itself as markup language.

Resources:

- <http://yaml.org/>
- <https://en.wikipedia.org/wiki/YAML>
- <http://www.yamllint.com/>

```
1 import os
2 import sys
3 import yaml
4
5 try:
6     yamlFilename = os.sys.argv[1]
7     yamlFile = open(yamlFilename, "r")
8 except:
9     print("filename does not exist")
10    sys.exit()
11 try:
12    yaml.load(yamlFile.read())
13 except:
14    print("YAML file is not valid.")
```

Figure 50.1: Python YAML example

50.2 JSON

The term JSON stand for *JavaScript Object Notation*. It is targeted as an open-standard file format that emphasizes on integration of human-readable text to transmit data objects. The data objects contain attribute value pairs. Although it originates from JavaScript, the format itself is language independent. It uses bracketest to allow organization of the data. Please note that YAML is a superset of JSON and not all YAML documents can be converted to JSON. Furthermore JSON does not support comments. For these reasons we often prefer to use YAMI instead of JSON. However JSON data can easily be translated to YAML as well as XML.

Resources:

- <https://en.wikipedia.org/wiki/JSON>
- <https://www.json.org/>

50.3 XML

XML stands for *Extensible Markup Language*. XML allows to define documents with the help of a set of rules in order to make it machine readable. The emphasize here is on machine readable as document in XML can become quickly complex and difficult to understand for humans. XML is used for documents as well as data structures.

A tutorial about XML is available at

- <https://www.w3schools.com/xml/default.asp>

Resources:

- <https://en.wikipedia.org/wiki/XML>




51. NoSQL

51.1 RDBMS vs. NoSQL

A discussion of relational database management systems (RDBMS) compared to NoSQL data storage systems is presented. This discussion includes an evolution of data storage systems, limitations with RDBMS in terms of scalability and how NoSQL fits into the picture in terms of big data.

[RDBMS vs. NoSQL \(9:22\)](#) 

[RDBMS vs. NoSQL \(Page 1\)](#) 

[RDBMS vs. NoSQL - pptx \(Page 1\)](#) 

51.2 NoSQL Characteristics

Clouds have arisen as an answer to the data demands of social media. Three major programs for NoSQL are BigTable, Dynamo, and CAP theory. More recently mongoDB has emerged as a leader in NoSQL databases and will also be discussed. NoSQL is not meant to replace SQL, but to tackle the large-data problems SQL is not well equipped to handle. SQL ACID transactions are Atomic, Consistent, Isolated, and Durable. Consistency can be either strong (ACID) or weak (BASE). CAP theorem offers Consistency, Availability, and Partition tolerance, only two of which can coexist for a shared-data system. NoSQL comes in two varieties, each with pros and cons: Key-Value or schema-less. Common advantages of NoSQL include their being open source and fault tolerant. The number of NoSQL databases makes it impossible to simply split them into Key-Value and schema-less categories, instead they can generally be placed in one of the three categories being document model, graph model and key-value and wide column models.

NoSQL Characteristics (10:31) 

NoSQL Characteristics (Page 11) 

NoSQL Characteristics - pptx (Page 11) 

51.2.1 Document Model

Relational databases store data in rows and columns, the class of NoSQL databases that fall into the document model store data as documents. Typically these documents are in JSON format and provide a straight forward way to model data that parallels object oriented programming where each document is an object. Each document stores one or more fields and within the field a value is stored in the form of an array, string date etc. In comparison to relational databases that distribute the records across columns that are connected with keys the document model stores records and their associated data in a single document. A key advantage to this model is that it reduces the need for JOIN operations that can be computationally expensive. Two examples of document databases are MongoDB and CouchDB.

51.2.2 Graph Model

As the name implies graph databases exploit the properties of graph structures and represent data through nodes and edges. Graph databases are useful for exploring the relationships of the components that make up an application by representing the data as a network of relationships. Graph databases can be useful for exploring social network connections, topologies of networks and supply chains. Two examples of graph databases are Neo4j and Graph.

51.2.3 Key-Value and Wide Column Models


Key-value stores are the most basic type on NoSQL databases as every item is stored as an attribute name or key paired with its value. In this model the value is not interpretable by the system as data can only be queried by the key. Key-value type databases do not enforce a set schema for key-value pairs making these database useful for unstructured and polymorphic data. In contrast to key-value stores, wide-column store data in distributed variable-dimensional sorted map where each record is stored as columns. These columns can be grouped into families and or columns can be distributed to several column families. Each column family has a primary key that can be queried in order to retrieve the data.

- <https://www.mongodb.com/collateral/top-5-considerations-when-evaluating-nosql-databases>

51.3 BigTable

Big Table is a key-value NoSQL model with data arranged in rows and columns. It is composed of Data File System, Chubby, and SSTable. A tablet is a range of rows in BigTable. The master node assigns tablets to tablet servers and manages these servers. Memory is conserved by making SSTables and memtables compact. BigTable is used in features of Google like their search engine and Google Earth.


BigTable (6:55) 

BigTable (Page 28) 

BigTable - pptx (Page 28) 

51.4 HBase

HBase is a NoSQL core component of the Hadoop Distributed File System. It is a scalable distributed data store. A timeline of HBase and Hadoop is shown. BigTable still has its uses but does not scale well to large amounts of analytic processing. HBase has a row-column structure similar to BigTable as well as master and slave nodes. Its place in the architecture of HDFS is shown in a diagram.

HBase (7:37) 

HBase (Page 44) 


HBase - pptx (Page 44) 

51.5 HBase Coding

This video gives an overview of the code used in the installation of HBase and connecting to it.

4:30 (HBase Coding) 

HBase Coding (Page 60) 

HBase Coding - pptx (Page 60) 

51.6 Draft: MongoDB

MongoDB belongs to the document model described above. Within a typical MongoDB server there are often multiple databases. Each database is a physical container of collections with its own set of files within the file system. A collection is a set of MongoDB documents similar to a typical relational database table. In MongoDB databases the collections are not required to follow a schema allowing documents in the same collection to have different fields. Some of the main advantages of MongoDB are its ease of scalability, lack of complex joins, schema-less, use of internal memory and the support for dynamic queries.

- https://www.tutorialspoint.com/mongodb/mongodb_quick_guide.htm

51.7 Indexing Applications

The setup of a search engine is briefly discussed using a diagram that contains the core components of a search engine. Google's search engine contains three key technologies: Google File System, BigTable, and MapReduce. However, research into big data remains difficult owing to the scope of its size. Social media data in particular is a huge source of data with numerous subsets, all of which demands specific approaches in terms of search queries. There are three stages to this approach: query, analysis, and visualization.

Indexing Applications (9:33) 

Indexing Applications (Page 1) 


[Indexing Applications - pptx \(Page 1\)](#) 

51.8 Related Work

Indexing improves efficiency in querying data subsets and analysis. Indices can be single (B+, Hash) or multi-dimensional (R, Quad). Four databases which utilize indexing are HBase, Cassandra, Riak, and MongoDB. Current indexing strategies have limits; for instance, they cannot support range queries or only retrieve Top ‘n’ most relevant topics. Customizability of indexing among NoSQL databases is desirable.

[Related Work \(5:56\)](#) 


[Related Work \(Page 11\)](#) 

[Related Work - pptx \(Page 11\)](#) 

51.9 Indexexamples

Mapping between metadata and raw index data is the essential issue with indexing. Examples are shown for HBase, Riak, and MongoDB. An abstract index structure contains index keys, entry IDs among multiple entries, and additional fields. Index configuration allows for customizability through choice of fields, which can be anything from timestamps, text, or retweet status.

[Indexexamples \(8:35\)](#) 


[Indexexamples \(Page 15\)](#) 

[Indexexamples - pptx \(Page 15\)](#) 

51.10 Indexing 101

User-defined index allows a user to select the fields used in their search. Data records are indexed or un-indexed. Index structure is made up of key, entry ID, and entry fields. A walk-through customized index creation is shown on HBase, called IndexedHBase. HBase is suited to accommodate the creation of index tables. A performance test of IndexedHBase is done on the Truthy Twitter repository, displaying the various tables that can be created with different criteria. Loading time for large-scale historical data can be reduced by adding nodes. Streaming data can be handled by increasing loaders. A comparison of query evaluation is made between IndexedHBase and Riak, with Riak being more efficient with small data loads but IndexedHBase proving superior for large-scale data.

[Indexing 101 \(9:53\)](#) 

[Indexing 101 \(Page 20\)](#) 

[Indexing 101 - pptx \(Page 20\)](#) 

51.11 Social Media Searches

The Truthy Project archives social media data by way of metadata memes. Some problems faced in analyzing this data include its large volume, sparsity of information in tweets, and attempting to arrange streaming tweets. Apache Open Stack upgrades Hadoop 2.0 with YARN and a new HDFS. A diagram displays an indexing setup for social media data with YARN.

Social Media Searches (6:19) 

Social Media Searches (Page 28) 

Social Media Searches - pptx (Page 28) 

51.12 Analysis Algorithms

Another method of use for inverted indices is in analysis algorithms. The mathematics involved in this is explored, as well as how it relates to index data, mapping, and reducing. Rather than scanning all raw data present, indices allow for searching only the relevant data. An example is given illustrating how this decreases the time needed to search hashtags in Twitter.

Analysis Algorithms (6:57) 

Analysis Algorithms (Page 35) 

Analysis Algorithms - pptx (Page 35) 



Computing with Raspberry Pi

52	Operating Systems	559
53	PI Software	561
53.1	Editors	
53.2	Python 3	
53.3	Python IDLE	
53.4	Docker	
53.5	Go	
54	Computing	565
54.1	Numpy	
54.2	Scipy	
54.3	Image Processing	
54.4	DHCP Server	
54.5	Gregor's Notes	



52. Operating Systems

In this chapter we will provide information in how to install additional software that may not be provided with the default operating system

You can run different operationg systems on the PI, this includes rasbian, NOOBS, Dexter (for Grove PI) but aslo Windows 10.

We like to get feedback on what OS is best suited for which task.

Rasbian

[Exercise 52.1](#) provide a mini tutorial

NOOBS

[Exercise 52.2](#) provide a mini tutorial

Dexter

[Exercise 52.3](#) provide a mini tutorial

Other Linux

[Exercise 52.4](#) provide a mini tutorial

52.0.1 Operatiing Systems

You can run different operationg systems on the PI, this includes rasbian, NOOBS, Dexter (for Grove PI) but aslo Windows 10.

We like to get feedback on what OS is best suited for which task.

Rasbian

Exercise 52.5 provide a mini tutorial ■

NOOBS

Exercise 52.6 provide a mini tutorial ■

Dexter

Exercise 52.7 provide a mini tutorial ■

Other Linux

Exercise 52.8 provide a mini tutorial ■

Windows IOT

Previously we played a bit with Windows on Raspberry, but it was not very well supported. This may have changed by now. Your task will be to evaluate its feature and contrast it to other OSes.

There seems to be support on developing PI application on WIndows 10 itself, but as we do not have windows machines, we have not tried this. You could explore and update us.

Exercise 52.9 provide a mini tutorial ■



53. PI Software

53.1 Editors

53.1.1 emacs

Exercise 53.1 provide a mini tutorial ■

53.1.2 vim

Exercise 53.2 provide a mini tutorial ■

53.1.3 gedit

Exercise 53.3 provide a mini tutorial ■

53.1.4 ssh

The most important program for us. describe how to enable without GUI
point to our other ssh section.

Exercise 53.4 make sure to describe how to not copy private keys !!! and instead use ssh
keygen on each machine, gather all pub keys and distribute
write a cloudmesh command for this (may already exist) ■

Exercise 53.5 provide a mini tutorial ■

53.2 Python 3

Exercise 53.6 provide a mini tutorial

sudo apt-get install python3

update to current version and try out. measure time it takes to do that, it may take a long time use time commands before and after

make sure you use apt install

- <https://liftcodeplay.com/2017/06/30/how-to-install-python-3-6-on-raspbian-linux-for-raspberry-pi/>

can we remove the source after install to save space?

53.3 Python IDLE

Click Menu > Programming > Python 3 (IDLE), and you'll get a new window called 'Python 3.4.2 Shell:'. This Shell works just like Python on the command line. Enter `print("Hello World")` to see the message.

Exercise 53.7 provide a mini tutorial

53.4 Docker

see go, maybe there are other better resources.

Exercise 53.8 provide a mini tutorial

53.5 Go

<https://blog.alexellis.io/golang-docker-rpi/>

Exercise 53.9 provide a mini tutorial

Exercise 53.10 provide a mini tutorial

53.5.1 Eclipse

Exercise 53.11 provide a mini tutorial

53.5.2 Adafruit Web IDE

- <https://learn.adafruit.com/webide/overview>

Exercise 53.12 provide a mini tutorial

53.5.3 Coder

- <https://googlecreativelab.github.io/coder/>

Exercise 53.13 provide a mini tutorial



54. Computing

54.1 Numpy

Refer to other section in book and describe what is different

Exercise 54.1 provide a mini tutorial

54.2 Scipy

Refer to other section in book and describe what is different

Exercise 54.2 provide a mini tutorial

54.3 Image Processing

Refer to other section in book and describe what is different

Exercise 54.3 provide a mini tutorial

54.4 DHCP Server

- <http://www.noveldevices.co.uk/rp-dhcp-server>

54.5 Gregor's Notes

This section contains some unordered notes by Gregor that you may want to look at and integrate into proper sections:

54.5.1 editor

find an editor that is installed and can be used on commandline nano, vi, other?

54.5.2 hostname

The hostname is stored in `/etc/hostname`. Edit the file and change it to a name such as green00, green01, green02, green03, green04, green05. Be consistent with the names. The 00 host should be the top most host in the cluster.

edit

```
1 nano /etc/hostname
```

after you edited the hostname

```
1 sudo /etc/init.d/hostname.sh start
```

Ideally we want to find out how to write the hostname after we burn the SDcard on the laptop that does the burning

develop a python script to do that

54.5.3 Gather the mac addresses

Is there a better way?

```
1 /sys/class/net/<interface-name>/address
2
3 cat /sys/class/net/eth0/address
4 cat /sys/class/net/wlan0/address
5 ifconfig eth0
```

develop a python script to do that

54.5.4 Enable sshd

Not tested

```
1 sudo mv /boot/boot_enable_ssh.rc /boot/boot.rc
2 sudo reboot
```

- <http://www.noveldevices.co.uk/rp-ssh>

NOt sure if this is needed:

“ You may find that you can connect to your Pi with SSH but the session hangs after a successful logon. This is usually caused because of a network QoS mismatch that affects certain switches and routers but you can correct this by editing the two files

```
1 /etc/ssh/ssh_config
2 /etc/ssh/sshd_config
```

and adding

```
1 IPQoS 0x00
```

to each file as the last record. “ develop a python script to do that

54.5.5 Wireless

Do not use iu secure. Do not leave your passwd on the pi as insecure.

```
1 sudo nano /etc/network/interfaces
2
3 auto wlan0
4 allow-hotplug wlan0
5 iface wlan0 inet dhcp
6 wpa-ssid "your-WLAN-SSID"
7 wpa-psk "your-WLAN-password"
```

develop a pthon script to do that

54.5.6 Update when on network

for now dont put pi on network so do this once we figure out a better way

```
1 sudo apt-get update
2 sudo apt-get install emacs
```

54.5.7 USB stick

```
1 cat /var/log/messages
```

find sda*

Make sure to find the right name.

```
1 sudo fdisk /dev/<device-name>
2 sudo mkfs -t vfat /dev/<device-name>
3 mkdir ~/<mount-point>
4 sudo mount /dev/<device-name> ~/<mount-point>
```

54.5.8 Locale

set locale to US so you can use keyboard

```
1 raspi-config
```

or

```
1 sudo dpkg-reconfigure locales
```

develop a python script to do that

54.5.9 DHCP server on 00

```
1 sudo apt-get update
2 sudo apt-get install isc-dhcp-server
3 sudo nano /etc/network/interfaces
```

Change it to

```
1 iface eth0 inet static
2 address <the-IP-address-of-your-Pi-that-will-be-the-DHCP-server>
3 netmask <the-subnet-mask-of-your-LAN>
4 gateway <the-IP-address-of-your-LAN-gateway>
5 sudo nano /etc/dhcp/dhcpd.conf
```

uncomment the info so the server can start

```

1 subnet <starting-IP-address-of-your-network> netmask <starting-IP-address-↔
  ↪ of-your-network> {
2
3     range <first-IP-address-of-your-DHCP-address-range> <last-IP-address-↔
  ↪ of-your-DHCP-address-range>;
4
5     option routers <the-IP-address-of-your-gateway-or-router>;
6
7     option broadcast-address <the-broadcast-IP-address-for-your-network>;
8
9 }
```

edit /etc/default/isc-dhcp-server

```

1 DHCPD_CONF=/etc/dhcp/dhcpd.conf
2 DHCPD_PID=/var/run/dhcpd.pid
3 INTERFACES="eth0"
```

```
1 sudo service isc-dhcp-server restart
```

54.5.10 Temperature

```
1 cat /sys/class/thermal/thermal_zone0/temp
```

54.5.11 graphana

could be helpful to monitor cluster/clusters

- <https://github.com/grafana/grafana>
- <https://github.com/weaveworks/grafanalib>

there are many more, just search. we have not tested them example with yaml

- <https://github.com/jakubplichtha/grafana-dashboard-builder>

light scheme

in /etc/grafana/grafana.ini uncomment line and set

```
1 default_theme = light
```

Cloud Computing with Raspberry Pi

55	Pi Cluster Form Factor	571
55.1	NAS (1 Pi)	
55.2	ClusterHat (4 Zero + 1 Pi)	
55.3	Cluster Case With Cooling (5 Pi)	
55.4	Bitscope Case (40 Pi)	
55.5	Bitscope Cluster (144 Pi)	
55.6	Build Your Own 5 Node Pi Cluster	
55.7	Single Pi	
55.8	Small Pi Cluster	
56	Cluster Setup	579
56.1	Links	
56.2	SDCards	
56.3	System Preparation without Monitor	
56.4	Post configuration	
56.5	Addon Hardware	
57	Docker Containers	583
57.1	Installation	
57.2	Configuration	
57.3	Example Container Creation with Dockerfile	
57.4	Using the Container	
57.5	REST Services	
58	Kubernetes	585
58.1	Installation	
58.2	Configuration	
58.3	Example Container Creation with Dockerfile	
58.4	Using the Container in Kubernetes	
58.5	REST Services	
59	Docker Swarm	587
59.1	Installation	
59.2	Configuration	
59.3	Example Swarm program	
59.4	Using the Swarm	
59.5	REST Services	
60	Spark	589
60.1	Installation	
60.2	Configuration	
60.3	Example Spark program	
60.4	Using Spark	
60.5	REST Services	
61	Slurm	591
61.1	Installation	
61.2	Configuration	
61.3	Example MPI program	
61.4	Using the Batch Queue	
61.5	REST Services	



55. Pi Cluster Form Factor

In this chapter we will discuss a number of opportunities to build small scale compute and cluster resources using Raspberry Pi's.

This includes the following:

- a NAS server with one Raspberry Pi 3 (Section [55.1](#))
- a Cluster using 1 Raspberry Pi as master and 4 Raspberry Zeros as workers (Section [55.2](#))
- a Cluster with 5 Raspberry Pi's (Section [55.3](#)). A minimum of 3 is needed.
- a Cluster with 40 Raspberry Pi's (Section [55.4](#))
- a Cluster with 144 Raspberry Pi's (Section [55.5](#))
- our recommended 5 node Raspberry Pi Cluster (Section [55.6](#)). 3 nodes need to be used at minimum.

55.1 NAS (1 Pi)

Although a NAS is not really a compute cluster the Pi has used many times to build a Network Attached Storage (NAS) server. In this configuration a HDD is attached to the Raspberry and the network features of the Raspberry is used to access the disk drive via software installed on the PI that make this easily possible. Many tutorials exists on the Web that help setting up such a device.

We like to hear from you if you have successfully developed such a NAS and provide us with such links. Links that may help include:

- <https://hackmypi.com/NASpi.php>



Figure 55.1: clusterhat

55.2 ClusterHat (4 Zero + 1 Pi)

The smallest cluster we came across is actually a hybrid cluster in which 4 Pi zeros attached to a Raspberry Pi 3. This is achieved via an add on board to the Pi 3 allowing to plug in Pi Zeros:

- <https://clusterhat.com/>

The Cluster HAT (Hardware Attached on Top) allows to attach 4 Raspberry Pi Zeros via to be attached to a regular Raspberry PI 3 to simulate a small cluster.

According to the Web Site it supports the following features:

- USB Gadget Mode: Ethernet and Serial Console.
- Onboard 4 port USB 2.0 hub.
- Raspberry Pi Zeros powered via Controller Pi GPIO (USB optional).
- Individual Raspberry Pi Zero power controlled via * Controller Pi GPIO (I2C).
- Connector for Controller Serial Console (FTDI Basic).
- Controller Pi can be rebooted without interrupting power to Pi Zeros (network recovers on boot).

Links:

- <https://www.raspberrypi.org/magpi/clusterhat-review-cluster-hat-kit/>

clusterhat on raspberrypi.org

Although this setup seems rather appealing, the issue is with obtaining Pi Zeros for the regional price of \$5. Typically users can only buy one for that price and must pay shipping. To buy more one has to buy a kit for about \$20. However, for that amount of money it may just be worth while to get Pi 3's instead of zero's. Nevertheless the formfactor is rather appealing.

55.3 Cluster Case With Cooling (5 Pi)

Many instructions on the Web exist describing how to build clusters with 3 or more Pi's. One of the considerations that we have to think about is that we may run rather demanding applications on such clusters causing heat issues. To eliminate them we must provide proper cooling. In some cluster projects cooling is not adequately addressed. Hence we like to provide an example that discusses in detail how to add a fan and what the fan has for an impact on the temperature.

- <http://climbers.net/sbc/add-fan-raspberry-pi/>
- <http://climbers.net/sbc/diy-raspberry-pi-3-cluster/>

From the above Web page we find the following information as shown in Table 55.1. From the

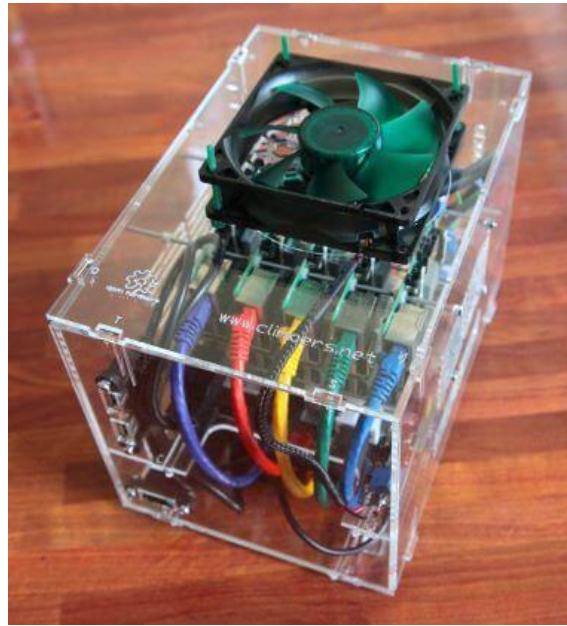


Figure 55.2:

data in the table it is clear that we need to keep the Pi from throttling while being in a case by adding a fan as obvious from experiment No. 2.

Table 55.1: Temperature comparison of fan impact

No.	Case	Fan	Direction	RPM	Idle	100% Load	Performance
1	no	no	-	-	41.0C	75.5C	OK (barely)
2	yes	no	-	-	45.0C	82.5C	throttles
3	yes	5V	in	unkown	37.9C	74.5C	OK (barely)
4	yes	7V	in	800	35.6C	69.5C	OK
5	yes	12V	in	1400	32.5C	61.1C	OK
6	yes	7V	out	800	34.5C	66.4C	OK

55.4 Bitscope Case (40 Pi)

A company from Australia called BitScope Designs offers a number of cases that leverage their Pi Blade boards allowing up to four Pis to be put together and sharing the same power supply. The blades are shown in Figure b.1. The rack to place 10 of them is shown in Figure b.2.

The cost of the blade rack is \$ 795.45 + \$60.00 shipping + import tax. This may originally sound expensive when compared to a single case, however as we can store 40 Pis in them and they can share the power-supply and reduce cabling we think this case is quite interesting overall due to its price-point of \$20 per Pi.

55.5 Bitscope Cluster (144 Pi)

<https://www.youtube.com/watch?v=78H-4KqVvrg>

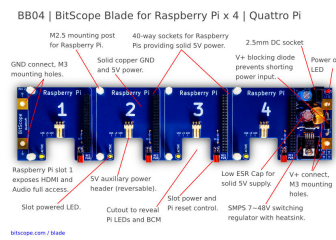


Figure 55.3: Bitscope blade for 4 Pi's.

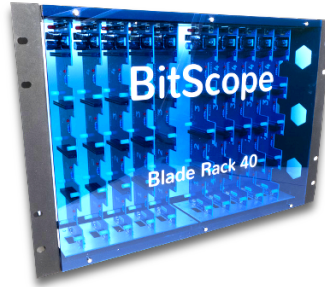


Figure 55.4: 40 Pi Blade rack.

Together with LANL a new cluster module that holds 144 Pis is developed. This system is targeted to be placed into a rack to create a large Pi cluster. The cost for such a module is about \$15K. Figure 55.5 shows the module and Figure 55.6 shows how multiple modules can be placed into a single rack.

55.5.1 Links

Additional information about this form factor can be found at the following links:

- <https://cluster.bitscope.com/solutions>
- <https://www.pcmper.com/news/General-Tech/BitScope-Unveils-Raspberry-Pi-Cluster-2880-CPU-Cores-LANL-HPC-RD>
- <http://my.bitscope.com/store/>
- <http://my.bitscope.com/store/?p=view&i=item+7>
- <http://www.newark.com/bitscope/bb04b/quattro-pi-board-raspberry-pi/dp/95Y0643>
- <http://linuxgizmos.com/rpi-expansion-boards-support-up-to-40-pi-clusters/>

55.6 Build Your Own 5 Node Pi Cluster

To experiment with building an elementary cluster one does not need to have a big budget. Such clusters are often dedicated to research tasks and are bound into security protocols that do not allow direct access. Instead it is possible to build such a cluster based on Raspberry PI's yourself if you are willing to spend the money or if you have access to PI's that you may loan from your department.

Table 55.2 lists one such possible parts list that will allow you to build a cluster for up to 5 nodes. However make sure to buy at least 3 Raspberry PI's with the appropriate memory. At minimum we recommend you get the 32GB SD card. We do not recommend any smaller as otherwise you



Figure 55.5: BitScope 144 cluster module.

- ✓ 1008 Nodes / Cabinet
- ✓ 42U Cabinet Height
- ✓ 37"~39" Cabinet Depth
- ✓ 7 x Modules / Cabinet
- ✓ 110V~240V or 24V DC
- ✓ 6kW Power Requirement
- ✓ Thermal Gradient Air Cooling Column Design
- ✓ 42 x 10GB SFP+ Fabric
- ✓ Blade Cluster Module 144

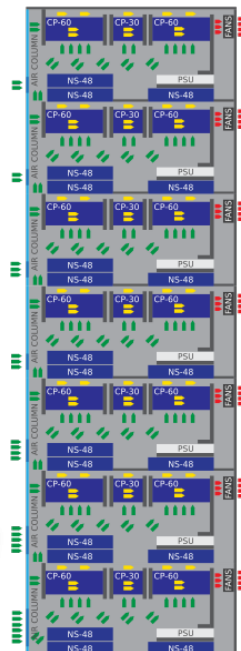
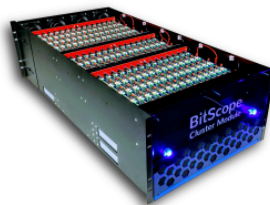


Figure 55.6: Rack placement of multiple BitScope 144 cluster modules.

will run out of memory. Additionally, you can add memory and disks on the USB ports. If you attach a HDD, make sure it has an external power supply and do not drive it from the USB power as otherwise the PI becomes unstable. A fan is at this time not yet included.

Naturally it is possible to modify the parts list and adapt. If you find better parts let us know. We have not included any case and you are welcome to share your suggestions with the class. For a case we are looking also for a good solution for a fan.

We suggest that when you build the cluster to do it on a table with a large white paper or board, or a tablecloth and take pictures of the various stages of the build so we can include it in this document.

Initially we just put rasbian as Operating system on the SD cards and test out each PI. To do so you will naturally need an SD card writer that you can hook up to your computer if it does not have one. As you will have to potentially do this more than once it is not recommended to buy an SD card with the OS on it. Buy the SD card writer instead so you can redo the flashing of the card when needed. In addition to the SD card you need a USB mouse and keyboard and a monitor or TV with HDMI port.

Locate setup instructions and write a tutorial in markdown that we will include here once it is finished. The tutorial is to be managed on github.

TODO: Class: check the network hub if it is a good choice as the one we originally chose could not be brought in sufficient quantity.

Table 55.2: Parts list for a Pi cluster, remember you need at least 3 Raspberry Pis

Price	Description	URL
\$29.99	Anker 60W 6-Port USB Wall Charger, PowerPort 6 for iPhone 7 / 6s / Plus, iPad Pro / Air 2 / mini, Galaxy S7 / S6 / Edge / Plus, Note 5 / 4, LG, Nexus, HTC and More	link
\$8.90	Cat 6 Ethernet Cable 1 ft White (6 Pack) - Flat Internet Network Cable - Jadaol Cat 6 Computer Cable short - Cat6 Ethernet Patch Lan Cable With	link
(1) \$19.99	D-link 8-Port Unmanaged Gigabit Switch (GO-SW-8G)	link
\$10.49	SanDisk Ultra 32GB microSDHC UHS-I Card with Adapter, Grey/Red, Standard Packaging (SDSQUNC-032G-GN6MA)	link
\$8.59	Short USB Cable, OKRAY 10 Pack Colorful Micro USB 2.0 Charging Data Sync Cable Cord for Samsung, Android Phone and Tablet, Nexus, HTC, Nokia, LG, Sony, Many Digital Cameras-0.66ft (7.87 Inch)	link
\$7.69	50 Pcs M2 x 20mm + 5mm Hex Hexagonal Threaded Spacer Support	link
\$7.99	Easycargo 15 pcs Raspberry Pi Heatsink Aluminum + Copper + 3M 8810 thermal conductive adhesive tape for cooling cooler Raspberry Pi 3, Pi 2, Pi Model B+	link
\$34.49	Raspberry Pi 3 Model B Motherboard (you need at least 3 of them)	link
(2) \$59.99	1TB drive	link
\$15.19	64GB flash	link
\$6.99	HDMI Cable, Rankie 2-Pack 6FT Latest Standard HDMI 2.0 HDTV Cable - Supports Ethernet, 3D, 4K and Audio Return (Black) - R1108	link
\$12.99	AUKEY USB C Adapter, USB C to USB 3.0 Adapter Aluminum 2 Pack for Samsung Note 8 S8 S8+, Google Pixel 2 XL, MacBook Pro, Nexus 6P 5X, LG G5 V20 (Gray)	link
\$19.19	For Raspberry Pi 3 2 TFT LCD Display, kuman 3.5 Inch 480x320 TFT Touch Screen Monitor for Raspberry Pi Model B B+ A+ A Module SPI Interface with Touch Pen SC06	link

(1) items were replaced with similar item

(2) item was not available

Exercise 55.1 In case you do not have access to multiple PIs conduct the Single Pi experiment.

■

55.7 Single Pi

You have been presented in Section 55.3 with a table that compares temperatures. Your task is to identify issues with the experiment and the table. Furthermore we like you to rerun a temperature experiment in the entire class.

TODO: Gregor: Add IoT section and PI configuration

1. Get a Pi3 model B, an HDMI cable, a power supply, a case. Such a configuration is listed in the IoT section.
2. Buy or manufacture a case of your choice. You can use a 3d-printer if you have one available.
3. Conduct a temperature experiment.

Discussion of these assignment is to be executed openly in class. Points will be issued only once the class agrees upon an experiment.

This exercise is not only to learn about the behaviour of the Pi, but also about how to coordinate experiments with a large number of students.

Exercise 55.2 What temperature measurement is missing from the table.

■

Exercise 55.3 How would you create an experiment under *load*.

■

Exercise 55.4 How would you create an experiment to which all students in different classes could contribute their values? Can the cloud be used?

■

Exercise 55.5 Collect the information from all class members using cloud services.

■

Exercise 55.6 Identify how to use the VPN server so you can use your Laptop instead of a TV or computer monitor. Write a tutorial.

■

55.8 Small Pi Cluster

In this set of exercises we will be building a small Raspberry Pi cluster. All of you will have to do Exercise ??, as well as one of the tasks related to Swarm, kubernetes or Spark.

It is important that you write down all steps very carefully as you are expected to use the steps to develop an automated deployment. For your cluster. Your tutorial will be tested by other groups and easy of installation completeness, and correctness will be evaluated. Teams that find issues and improve deployment tutorials will receive points. TA's will also replicate these steps to identify a fair evaluation without bias.

Exercise 55.7 Build groups of up to 5 people. Make a plan on what needs to be done to build the cluster and develop a schedule. Include in this plan (a) obtaining the material the hardware build, (b) the installation of the operating system (c) the testing of the system (d) familiarizing with the OS.

■

Exercise 55.8 Install a docker Swarm cluster on your PI. Develop a tutorial in markdown and mind plagiarism. Contribute your tutorial to this document to get acknowledged and credit. Work with others in class to coordinate a single tutorial. ■

Exercise 55.9 Install a kubernetes cluster on your PI. Develop a tutorial in markdown and mind plagiarism. Contribute your tutorial to this document to get acknowledged and credit. Work with others in class to coordinate a single tutorial. ■

Exercise 55.10 Install a spark cluster on your PI. Develop a tutorial in markdown and mind plagiarism. Contribute your tutorial to this document to get acknowledged and credit. Work with others in class to coordinate a single tutorial. ■

55.8.1 Virtual Raspberry Cluster

It should also be possible to create a virtual raspberry PI cluster while for example using virtual box. This requires two steps. First the deployment of a virtualized Raspberry PI. The following information may be useful for this

- <http://dbakevlar.com/2015/08/emulating-a-raspberry-pi-on-virtualbox/>

The next step includes the deployment of multiple VMs emulating Raspberrys. Naturally each should have its own name so you can distinguish them. INstead of just using the GUI, it would be improtant to find out how to start them from a commandline as a shell script as well as tear them down.

Next you will need to make sure you can communicate from the PIs to each other. This is naturally the same as on a real cluster

Exercise 55.11 provide a tutorial ■

This can be chosen as part of your project, but you need to develop a cloudmesh command for managing the cluster. This includes starting and stoping as well as checkpointing the cluster from a cloudmesh command. Furthermore you need to benchmark it and identify how to do this and contrast this to other clusters that you may start or have access to. Please get in contact with Gregor. THis prject is reserved for online students, as residential students will have access to real Raspberry PI hardware.



56. Cluster Setup

In this section we discuss how we setup the cluster. To explore that we can conduct the setup without monitor, we are collecting in this section a variety of tasks in regards to it and documenting solutions for them. Students in the residential or in the online classes that participate in cluster building can contribute to this section.

A number of students have been assigned for particular tasks. and will be added here by the TA's. It is expected that you conduct your task ASAP.

56.1 Links

A number of useful links may help for this task. However be aware that we want to develop a *script* for most of the tasks eliminating input by hand. The information here deals to identifying the information needed to do so.

- <https://hackernoon.com/raspberry-pi-headless-install-462ccabd75d0>
- <https://installvirtual.com/enable-ssh-in-raspberry-pi-without-monitor/>

According to

- <https://www.raspberrypi.org/documentation/configuration/raspi-config.md>

raspi-config writes a file `/boot/config.txt`. Maybe it is a good idea to inspect this file manipulate it and see if there are fields we simply could write and overwrite it on the SDcard. See also

- https://elinux.org/R-Pi_configuration_file

56.2 SDCards

The cluster will have an SDCard in each of the PIs. To enable them to boot an operating system has to be put on the card. We recommend that you use Etcher for doing so which is available for Linux, Windows, and OSX. Etcher can be found at

- <https://etcher.io/>

If there are better tools or methods please document.

Exercise 56.1 For your OS, discuss how to install Etcher. Then discuss how to burn the OS on the SD card. ■

56.2.1 Linux

TODO: Arnav sp18-503

Exercise 56.2 Describe this for Linux. ■

56.2.2 OSX

TODO: Yue Guo sp18-508, Min Chen sp18-405

Exercise 56.3 Describe this for OSX. ■

56.2.3 Windows 10

TODO: Keerthi sp18-602

Exercise 56.4 Describe this for Windows 10. ■

56.2.4 Creating Backup

TODO: Yue sp18-508

Exercise 56.5 In this task you will be documenting how to create a backup of the image on the SDCard. ■

56.2.5 Duplication

TODO: Keerthi sp18-602

Exercise 56.6 Let us assume you have installed a lot of great programs on the SDCard. In a cluster, we need to duplicate this card for each PI in the cluster. Is there a way for us to duplicate the software ■

56.2.6 Mount the SD Card on the Host System

Exercise 56.7 Describe on how to mount the SDCard on the host system so you can manipulate the files on the SDCard ■

56.2.7 Linux

TODO: Arnav sp18-503

Exercise 56.8 Describe this for Linux. ■

56.2.8 OSX

TODO: Yue Guo sp18-508, Min Chen sp18-405

Exercise 56.9 Describe this for OSX. ■

56.2.9 Windows 10

TODO: Keerthi sp18-602

Exercise 56.10 Describe this for Windows 10. ■

56.3 System Preparation without Monitor

56.3.1 hostname

Exercise 56.11 Find a way on how to name the host as each of the cluster nodes will need its own unique name. We simple use color01, where color is the color of the USB cables you have and the number is the ith PI starting from the top ■

56.3.2 SSH

TODO: Goutham sp18-401

Exercise 56.12 Describe how you enable SSH without a monitor ■

56.3.3 key

TODO: Priyadarshini sp18-421, Ankita sp18-502

Exercise 56.13 Describe how you can generate a private key at the right location on the SDCard. Place your own public key on the SDCard
Write python programs for this. ■

56.3.4 password

Exercise 56.14 Describe how you can change the password on the SDCard ■

TODO: Surya sp18-418

56.4 Post confoguration

56.4.1 Network Addresses

Exercise 56.15 Find a way to find all the network addresses from all PIs attached to the network switch. ■

56.4.2 key

TODO: Priyadarshini sp18-421, Ankita sp18-502

Exercise 56.16 Write a python program that does the following:

- (1) login with ssh on each PI and call ssh-keygen to generate a unique key on each PI.
- (2) copy this .pub keys to your computer and store them into a file called authorized_keys
- (3) copy that file to all Pis
- (4) you may also have to copy your authorized key file to

56.4.3 VNC

Exercise 56.17 find out how to set up vnc so you can login into the PI and see its GUI ■

- <https://www.raspberrypi.org/forums/viewtopic.php?t=74176>
- <https://www.raspberrypi.org/documentation/remote-access/vnc/>

```

1
2 alias IP=<IPADDRESSOFPI>
3
4 sh pi@$IP
5 pi@IP's password:
6 Linux raspberrypi 3.10.25+ #622 PREEMPT Fri Feb 3 20:00:00 GMT 2018 armv6l
7 pi@raspberrypi ~ $ sudo apt-get tightvncserver # download the VNC server
8 pi@raspberrypi ~ $ tightvncserver # start the VNC server
9 pi@raspberrypi ~ $ vncserver :0 -geometry 1920x1080 -depth 24 # start a ↔
   ↔ VNC session
10
11 Now, back on your Mac:
12 In the Finder, select Go => Connect To Server...
13 Type vnc://xxx.xxx.xxx.xxx (where xxx.xxx.xxx.xxx is the IP address that ↔
   ↔ you discovered in step 2.
14 Click [Connect]. This will launch the Screen Sharing application, and
15 with a little luck, you should see this image.
```

56.5 Addon Hardware

recently ordered add on hardware

power switches

- https://www.amazon.com/dp/B01DE57SD4/ref=psdc_6396124011_t2_B075WZJL6N

SDCard Writers

- https://www.amazon.com/Collection-MicroSD-MicroSDHC-MicroSDXC-Kingston/dp/B01IF7TPMS/ref=sr_1_15?s=electronics&ie=UTF8&qid=1518271583&sr=1-15&keywords=Micro+SD+Card+writer



57. Docker Containers

57.1 Instalation

[Exercise 57.1](#) provide a tutorial

57.2 Configuration

[Exercise 57.2](#) provide a tutorial

57.3 Example Container Creation with Dockerfile

[Exercise 57.3](#) provide a tutorial

57.4 Using the Container

[Exercise 57.4](#) provide a tutorial

57.5 REST Services

[Exercise 57.5](#) develop OpenAPI REST services to interact with the cluster



58. Kubernetes

TODO: Tim sp18-526, Juliano sp18-601

58.1 Instalation

Exercise 58.1 provide a tutorial

58.2 Configuration

Exercise 58.2 provide a tutorial

58.3 Example Container Creation with Dockerfile

Exercise 58.3 provide a tutorial

58.4 Using the Container in Kubernetes

Exercise 58.4 provide a tutorial

58.5 REST Services

Exercise 58.5 develop OpenAPI REST services to interact with the cluster



59. Docker Swarm

TODO: Hao Tian sp18-524, Lin Qingyun sp18-515

59.1 Instalation

Exercise 59.1 provide a tutorial ■

59.2 Configuration

Exercise 59.2 provide a tutorial ■

59.3 Example Swarm program

Exercise 59.3 provide a tutorial ■

59.4 Using the Swarm

Exercise 59.4 provide a tutorial ■

59.5 REST Services

Exercise 59.5 develop OpenAPI REST services to interact with the cluster ■



60. Spark

TODO: Karan Kotabagi sp18-412, Manoj, Min sp18-405, Ramya sp18-406, Karan Kamatagi sp18-410

60.1 Instalation

[Exercise 60.1](#) provide a tutorial

60.2 Configuration

[Exercise 60.2](#) provide a tutorial

60.3 Example Spark program

[Exercise 60.3](#) provide a tutorial

60.4 Using Spark

[Exercise 60.4](#) provide a tutorial

60.5 REST Services

[Exercise 60.5](#) develop OpenAPI REST services to interact with the cluster



61. Slurm

This may be inspiring

- <https://github.com/ajdecon/ansible-pi-cluster/blob/master/README.md>

61.1 Instalation

Exercise 61.1 provide a tutorial ■

61.2 Configuration

Exercise 61.2 provide a tutorial ■

61.3 Example MPI program

Exercise 61.3 provide a tutorial ■

61.4 Using the Batch Queue

Exercise 61.4 provide a tutorial ■

61.5 REST Services

Exercise 61.5 develop OpenAPI REST services to interact with the cluster ■

Big Data Applications

62	Big Data Use Cases Survey	595
62.1	NIST Big Data Public Working Group	
62.2	51 Big Data Use Cases	
62.3	Features of 51 Big Data Use Cases	
63	Health Informatics	605
63.1	Big Data and Health	
63.2	Status of Healthcare Today	
63.3	Telemedicine (Virtual Health)	
63.4	Medical Big Data in the Clouds	
64	e-Commerce and LifeStyle	611
64.1	Recommender Systems	
64.2	Item-based Collaborative Filtering and its Technologies	
65	Physics	617
65.1	Looking for Higgs Particles	
65.2	SKA – Square Kilometer Array	
65.3	Radar	
66	Sensors	625
66.1	Internet of Things	
66.2	Robotics and IoT	
66.3	Industrial Internet of Things	
66.4	Sensor Clouds	
66.5	Earth/Environment/Polar Science data gathered by Sensors	
66.6	Ubiquitous/Smart Cities	
66.7	U-Korea (U=Ubiquitous)	
66.8	Smart Grid	
66.9	Resources	
67	Sports	629
67.1	Basic Sabermetrics	
67.2	Advanced Sabermetrics	
67.3	PITCHf/X	
67.4	Other Sports	
68	Web Search and Text Mining	633
68.1	Web Search and Text Mining	
68.2	Search Engines	
68.3	Topics in Web Search and Text Mining	



62. Big Data Use Cases Survey

This section covers 51 values of X and an overall study of Big data that emerged from a NIST (National Institute for Standards and Technology) study of Big data. The section covers the NIST Big Data Public Working Group (NBD-PWG) Process and summarizes the work of five subgroups: Definitions and Taxonomies Subgroup, Reference Architecture Subgroup, Security and Privacy Subgroup, Technology Roadmap Subgroup and the Requirements and Use Case Subgroup. 51 use cases collected in this process are briefly discussed with a classification of the source of parallelism and the high and low level computational structure. We describe the key features of this classification.

62.1 NIST Big Data Public Working Group

This unit covers the NIST Big Data Public Working Group (NBD-PWG) Process and summarizes the work of five subgroups: Definitions and Taxonomies Subgroup, Reference Architecture Subgroup, Security and Privacy Subgroup, Technology Roadmap Subgroup and the Requirements and Use Case Subgroup. The work of latter is continued in next two units.

45 (Overview) 

62.1.1 Introduction to NIST Big Data Public Working

The focus of the (NBD-PWG) is to form a community of interest from industry, academia, and government, with the goal of developing a consensus definitions, taxonomies, secure reference architectures, and technology roadmap. The aim is to create vendor-neutral, technology and infrastructure agnostic deliverables to enable big data stakeholders to pick-and-choose best analytics tools for their processing and visualization requirements on the most suitable computing platforms and clusters while allowing value-added from big data service providers and flow of data between

the stakeholders in a cohesive and secure manner.

Introduction (13:02) 

62.1.2 Definitions and Taxonomies Subgroup

The focus is to gain a better understanding of the principles of Big Data. It is important to develop a consensus-based common language and vocabulary terms used in Big Data across stakeholders from industry, academia, and government. In addition, it is also critical to identify essential actors with roles and responsibility, and subdivide them into components and sub-components on how they interact/ relate with each other according to their similarities and differences.

For Definitions: Compile terms used from all stakeholders regarding the meaning of Big Data from various standard bodies, domain applications, and diversified operational environments. For Taxonomies: Identify key actors with their roles and responsibilities from all stakeholders, categorize them into components and subcomponents based on their similarities and differences. In particular data Science and Big Data terms are discussed.

Taxonomies (7:42) 

62.1.3 Reference Architecture Subgroup

The focus is to form a community of interest from industry, academia, and government, with the goal of developing a consensus-based approach to orchestrate vendor-neutral, technology and infrastructure agnostic for analytics tools and computing environments. The goal is to enable Big Data stakeholders to pick-and-choose technology-agnostic analytics tools for processing and visualization in any computing platform and cluster while allowing value-added from Big Data service providers and the flow of the data between the stakeholders in a cohesive and secure manner. Results include a reference architecture with well defined components and linkage as well as several exemplars.

Architecture (10:05) 

62.1.4 Security and Privacy Subgroup

The focus is to form a community of interest from industry, academia, and government, with the goal of developing a consensus secure reference architecture to handle security and privacy issues across all stakeholders. This includes gaining an understanding of what standards are available or under development, as well as identifies which key organizations are working on these standards. The Top Ten Big Data Security and Privacy Challenges from the CSA (Cloud Security Alliance) BDWG are studied. Specialized use cases include Retail/Marketing, Modern Day Consumerism, Nielsen Homescan, Web Traffic Analysis, Healthcare, Health Information Exchange, Genetic Privacy, Pharma Clinical Trial Data Sharing, Cyber-security, Government, Military and Education.

Security (9:51) 

62.1.5 Technology Roadmap Subgroup

The focus is to form a community of interest from industry, academia, and government, with the goal of developing a consensus vision with recommendations on how Big Data should move forward by performing a good gap analysis through the materials gathered from all other NBD

[section/bigdata/applications/usecases.tex](#)

subgroups. This includes setting standardization and adoption priorities through an understanding of what standards are available or under development as part of the recommendations. Tasks are gather input from NBD subgroups and study the taxonomies for the actors' roles and responsibility, use cases and requirements, and secure reference architecture; gain understanding of what standards are available or under development for Big Data; perform a thorough gap analysis and document the findings; identify what possible barriers may delay or prevent adoption of Big Data; and document vision and recommendations.

Technology (4:14) 

62.1.6 Interfaces Subgroup

This subgroup is working on the following document: *NIST Big Data Interoperability Framework: Volume 8, Reference Architecture Interface*.

This document summarizes interfaces that are instrumental for the interaction with Clouds, Containers, and HPC systems to manage virtual clusters to support the NIST Big Data Reference Architecture (NBDRA). The Representational State Transfer (REST) paradigm is used to define these interfaces allowing easy integration and adoption by a wide variety of frameworks. . This volume, Volume 8, uses the work performed by the NBD-PWG to identify objects instrumental for the NIST Big Data Reference Architecture (NBDRA) which is introduced in the NBDIF: Volume 6, Reference Architecture.

This presentation was given at the *2nd NIST Big Data Public Working Group (NBD-PWG) Workshop* in Washington DC in June 2017. It explains our thoughts on deriving automatically a reference architecture form the Reference Architecture Interface specifications directly from the document.

The workshop Web page is located at

- <https://bigdatawg.nist.gov/workshop2.php>

The agenda of the workshop is as follows:

- https://bigdatawg.nist.gov/2017_NIST_Big_Data_PWG_WorkshopAgenda_with_Speakers_Bio.pdf

The Web cas of the presentation is given bellow, while you need to fast forward to a particular time

- Webcast: Interface subgroup: <https://www.nist.gov/news-events/events/2017/06/2nd-nist-big-data-public-working-group-nbd-pwg-workshop>
– see: Big Data Working Group Day 1, part 2 Time start: 21:00 min, Time end: 44:00
- Slides: <https://github.com/cloudmesh/cloudmesh.rest/blob/master/docs/NBDPWG-vol8.pptx?raw=true>
- Document: <https://github.com/cloudmesh/cloudmesh.rest/raw/master/docs/NIST.SP.1500-8-draft.pdf>

You are welcome to view other presentations if you are interested.

62.1.7 Requirements and Use Case Subgroup


The focus is to form a community of interest from industry, academia, and government, with the goal of developing a consensus list of Big Data requirements across all stakeholders. This includes gathering and understanding various use cases from diversified application domains. Tasks are gather use case input from all stakeholders; derive Big Data requirements from each use case; analyze/prioritize a list of challenging general requirements that may delay or prevent adoption

of Big Data deployment; develop a set of general patterns capturing the *essence* of use cases (not done yet) and work with Reference Architecture to validate requirements and reference architecture by explicitly implementing some patterns based on use cases. The progress of gathering use cases (discussed in next two units) and requirements systemization are discussed.

Requirements (27:28) 

62.2 51 Big Data Use Cases

This unit consists of one or more slides for each of the 51 use cases - typically additional (more than one) slides are associated with pictures. Each of the use cases is identified with source of parallelism and the high and low level computational structure. As each new classification topic is introduced we briefly discuss it but full discussion of topics is given in following unit.

100 (51) 

62.2.1 Government Use Cases

This covers Census 2010 and 2000 - Title 13 Big Data; National Archives and Records Administration Accession NARA, Search, Retrieve, Preservation; Statistical Survey Response Improvement (Adaptive Design) and Non-Traditional Data in Statistical Survey Response Improvement (Adaptive Design).

Government Use Cases (17:43) 

62.2.2 Commercial Use Cases

This covers Cloud Eco-System, for Financial Industries (Banking, Securities & Investments, Insurance) transacting business within the United States; Mendeley - An International Network of Research; Netflix Movie Service; Web Search; IaaS (Infrastructure as a Service) Big Data Business Continuity & Disaster Recovery (BC/DR) Within A Cloud Eco-System; Cargo Shipping; Materials Data for Manufacturing and Simulation driven Materials Genomics.

Commercial Use Cases (17:43) 

62.2.3 Defense Use Cases

This covers Large Scale Geospatial Analysis and Visualization; Object identification and tracking from Wide Area Large Format Imagery (WALF) Imagery or Full Motion Video (FMV) - Persistent Surveillance and Intelligence Data Processing and Analysis.

Defense Use Cases (15:43) 

62.2.4 Healthcare and Life Science Use Cases

This covers Electronic Medical Record (EMR) Data; Pathology Imaging/digital pathology; Computational Bioimaging; Genomic Measurements; Comparative analysis for metagenomes and genomes; Individualized Diabetes Management; Statistical Relational Artificial Intelligence for Health Care; World Population Scale Epidemiological Study; Social Contagion Modeling for Planning, Public Health and Disaster Management and Biodiversity and LifeWatch.

section/bigdata/applications/usecases.tex

Healthcare and Life Science Use Cases (30:11) 

62.2.5 Deep Learning and Social Networks Use Cases

This covers Large-scale Deep Learning; Organizing large-scale, unstructured collections of consumer photos; Truthy: Information diffusion research from Twitter Data; Crowd Sourcing in the Humanities as Source for Big and Dynamic Data; CINET: Cyberinfrastructure for Network (Graph) Science and Analytics and NIST Information Access Division analytic technology performance measurement, evaluations, and standards.

Deep Learning and Social Networks Use Cases (14:19) 

62.2.6 Research Ecosystem Use Cases

DataNet Federation Consortium DFC; The ‘Discinnet process’, metadata -big data global experiment; Semantic Graph-search on Scientific Chemical and Text-based Data and Light source beamlines.

Research Ecosystem Use Cases (9:09) 

62.2.7 Astronomy and Physics Use Cases

This covers Catalina Real-Time Transient Survey (CRTS): a digital, panoramic, synoptic sky survey; DOE Extreme Data from Cosmological Sky Survey and Simulations; Large Survey Data for Cosmology; Particle Physics: Analysis of LHC Large Hadron Collider Data: Discovery of Higgs particle and Belle II High Energy Physics Experiment.

Astronomy and Physics Use Cases (17:33) 

62.2.8 Environment, Earth and Polar Science Use Cases

EISCAT 3D incoherent scatter radar system; ENVRI, Common Operations of Environmental Research Infrastructure; Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets; UAVSAR Data Processing, DataProduct Delivery, and Data Services; NASA LARC/GSFC iRODS Federation Testbed; MERRA Analytic Services MERRA/AS; Atmospheric Turbulence - Event Discovery and Predictive Analytics; Climate Studies using the Community Earth System Model at DOE’s NERSC center; DOE-BER Subsurface Biogeochemistry Scientific Focus Area and DOE-BER AmeriFlux and FLUXNET Networks.

Environment, Earth and Polar Science Use Cases (25:29) 

62.2.9 Energy Use Case

This covers Consumption forecasting in Smart Grids.

Energy Use Case (4:01) 

62.3 Features of 51 Big Data Use Cases

This unit discusses the categories used to classify the 51 use-cases. These categories include concepts used for parallelism and low and high level computational structure. The first lesson is an introduction to all categories and the further lessons give details of particular categories.

[43 \(Features\)](#) 

62.3.1 Summary of Use Case Classification

This discusses concepts used for parallelism and low and high level computational structure. Parallelism can be over People (users or subjects), Decision makers; Items such as Images, EMR, Sequences; observations, contents of online store; Sensors -- Internet of Things; Events; (Complex) Nodes in a Graph; Simple nodes as in a learning network; Tweets, Blogs, Documents, Web Pages etc.; Files or data to be backed up, moved or assigned metadata; Particles/cells/mesh points. Low level computational types include PP (Pleasingly Parallel); MR (MapReduce); MRStat; MRIter (Iterative MapReduce); Graph; Fusion; MC (Monte Carlo) and Streaming. High level computational types include Classification; S/Q (Search and Query); Index; CF (Collaborative Filtering); ML (Machine Learning); EGO (Large Scale Optimizations); EM (Expectation maximization); GIS; HPC; Agents. Patterns include Classic Database; NoSQL; Basic processing of data as in backup or metadata; GIS; Host of Sensors processed on demand; Pleasingly parallel processing; HPC assimilated with observational data; Agent-based models; Multi-modal data fusion or Knowledge Management; Crowd Sourcing.

[Summary of Use Case Classification \(23:39\)](#) 

62.3.2 Database(SQL) Use Case Classification

This discusses classic (SQL) database approach to data handling with Search&Query and Index features. Comparisons are made to NoSQL approaches.

[Database \(SQL\) Use Case Classification \(11:13\)](#) 

62.3.3 NoSQL Use Case Classification

This discusses NoSQL (compared in previous lesson) with HDFS, Hadoop and Hbase. The Apache Big data stack is introduced and further details of comparison with SQL.

[NoSQL Use Case Classification \(11:20\)](#) 

62.3.4 Other Use Case Classifications

This discusses a subset of use case features: GIS, Sensors. the support of data analysis and fusion by streaming data between filters.

[Use Case Classifications I \(12:42\)](#) 

This discusses a subset of use case features: Pleasingly parallel, MRStat, Data Assimilation, Crowd sourcing, Agents, data fusion and agents, EGO and security.

[Use Case Classifications II \(20:18\)](#) 

[section/bigdata/applications/usecases.tex](#)

This discusses a subset of use case features: Classification, Monte Carlo, Streaming, PP, MR, MRStat, MRIter and HPC(MPI), global and local analytics (machine learning), parallel computing, Expectation Maximization, graphs and Collaborative Filtering.

Use Case Classifications III (17:25) 

TODO: These resources have not all been checked to see if they still exist this is currently in progress

62.3.5 Resources

- NIST Big Data Public Working Group (NBD-PWG) Process <https://www.nist.gov/el/cyber-physical-systems/big-data-pwg>
- Big Data Definitions: <http://dx.doi.org/10.6028/NIST.SP.1500-1> (link is external)
- Big Data Taxonomies: <http://dx.doi.org/10.6028/NIST.SP.1500-2> (link is external)
- Big Data Use Cases and Requirements: <http://dx.doi.org/10.6028/NIST.SP.1500-3> (link is external)
- Big Data Security and Privacy: <http://dx.doi.org/10.6028/NIST.SP.1500-4> (link is external)
- Big Data Architecture White Paper Survey: <http://dx.doi.org/10.6028/NIST.SP.1500-5> (link is external)
- Big Data Reference Architecture: <http://dx.doi.org/10.6028/NIST.SP.1500-6> (link is external)
- Big Data Standards Roadmap: <http://dx.doi.org/10.6028/NIST.SP.1500-7> (link is external)

Some of the links bellow may be outdated. Please let us know the new links and notify us of the outdated links.

- DCGSA Standard Cloud: <https://www.youtube.com/watch?v=l4Qii7T8zeg>
- On line 51 Use Cases <http://bigdatawg.nist.gov/usecases.php>
- Summary of Requirements Subgroup http://bigdatawg.nist.gov/_uploadfiles/M0245_v5_6066621242.docx
- Use Case 6 Mendeley <http://mendeley.com%20http://dev.mendeley.com>
- Use Case 7 Netflix <http://www.slideshare.net/xamat/building-largescale-realworld-recommender-systems-recsys2012-tutoria>
- Use Case 8 Search <http://www.slideshare.net/kleinerperkins/kpcb-internet-trends-2013>, http://webcourse.cs.technion.ac.il/236621/Winter2011-2012/en/ho_Lectures.html, <http://www.ifis.cs.tu-bs.de/teaching/ss-11/irws>, <http://www.slideshare.net/bee chung/recommender-systems-tutorialpart1intro>, <http://www.worldwidewebsite.com/>
- Use Case 9 IaaS (Infrastructure as a Service) Big Data Business Continuity & Disaster Recovery (BC/DR) Within A Cloud Eco-System provided by Cloud Service Providers (CSPs) and Cloud Brokerage Service Providers (CBSPs) <http://www.disasterrecovery.org/>
- Use Case 11 and Use Case 12 Simulation driven Materials Genomics <https://www.materialsproject.org/>
- Use Case 13 Large Scale Geospatial Analysis and Visualization <http://www.opengeospatial.org/standards>, <http://geojson.org/>, <http://earth-info.nga.mil/publications/specs/printed/CADRG/cadrg.html>
- Use Case 14 Object identification and tracking from Wide Area Large Format Imagery (WALF) Imagery or Full Motion Video (FMV) - Persistent Surveillance <http://www.>

- militaryaerospace.com/topics/m/video/79088650/persistent-surveillance-relies-on-extracting-relevant-data-points-and-connecting-the-dots.htm, <http://www.defencetalk.com/wide-area-persistent-surveillance-revolutionizes-tactical-isr-45745/>
- Use Case 15 Intelligence Data Processing and Analysis http://www.afcea-aberdeen.org/files/presentations/AFCEAAberdeen_DCGSA_COLWells_PS.pdf, http://stids.c4i.gmu.edu/papers/STIDSPapers/STIDS2012/_T14/_SmithEtAl/_HorizontalIntegrationOfWarfi.pdf, http://stids.c4i.gmu.edu/STIDS2011/papers/STIDS2011_CR_T1_SalmenEtAl.pdf, <https://www.youtube.com/watch?v=l4Qii7T8zeg>, <http://dcgsa.apg.army.mil/>
 - Use Case 16 Electronic Medical Record (EMR) Data: Regenstrief Institute, Logical observation identifiers names and codes, Indiana Health Information Exchange, Institute of Medicine Learning Healthcare System
 - Use Case 17 Pathology Imaging/digital pathology; <https://web.cci.emory.edu/confluence/display/PAIS> , <https://web.cci.emory.edu/confluence/display/HadoopGIS>
 - Use Case 19 Genome in a Bottle Consortium: www.genomeinabottle.org
 - Use Case 20 Comparative analysis for metagenomes and genomes <http://img.jgi.doe.gov/>
 - Use Case 25 Biodiversity and LifeWatch
 - Use Case 26 Deep Learning: Recent popular press coverage of deep learning technology: <http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-intelligence.html> , <http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html> , http://www.wired.com/2013/06/andrew_ng/,
A recent research paper on HPC for Deep Learning: http://www.stanford.edu/~acoates/papers/CoatesHuvalWangWuNgCatanzaro_icml2013.pdf, Widely-used tutorials and references for Deep Learning: http://ufldl.stanford.edu/wiki/index.php/Main_Page, <http://deeplearning.net/>
 - Use Case 27 Organizing large-scale, unstructured collections of consumer photos <http://vision.soic.indiana.edu/projects/disco/>
 - Use Case 28 Truthy: Information diffusion research from Twitter Data <http://truthy.indiana.edu/> , <http://cnets.indiana.edu/groups/nan/truthy/> , <http://cnets.indiana.edu/groups/nan/despic/>
 - Use Case 30 CINET: Cyberinfrastructure for Network (Graph) Science and Analytics http://cinet.vbi.vt.edu/cinet_new/
 - Use Case 31 NIST Information Access Division analytic technology performance measurement, evaluations, and standards <http://www.nist.gov/itl/iad/>
 - Use Case 32 DataNet Federation Consortium DFC: [The DataNet Federation Consortium](http://www.datanet-federation.org/), iRODS
 - Use Case 33 The ‘Discinnet process’, metadata < - > big data global experiment <http://www.discinnet.org/>
 - Use Case 34 Semantic Graph-search on Scientific Chemical and Text-based Data http://www.eurekalert.org/pub_releases/2013-07/aiop-ffm071813.php , <http://xpdb.nist.gov/chemblast/pdb.pl>
 - Use Case 35 Light source beamlines <http://www-als.lbl.gov/> , <https://www1.aps.anl.gov/>
 - Use Case 36 CRTS survey, CSS survey ; For an overview of the classification challenges, see, e.g., <http://arxiv.org/abs/1209.1681>
 - Use Case 37 DOE Extreme Data from Cosmological Sky Survey and Simulations <http://www.lsst.org/lsst/> , <http://www.nersc.gov/> , <http://www.nersc.gov/assets/Uploads/HabibcosmosimV2.pdf>
 - Use Case 38 Large Survey Data for Cosmology <http://desi.lbl.gov/> , <http://www.darkenergysurvey.org/>

- Use Case 39 Particle Physics: Analysis of LHC Large Hadron Collider Data: Discovery of Higgs particle <http://grids.ucs.indiana.edu/ptliupages/publications/Where%20does%20all%20the%20data%20come%20from%20v7.pdf>, http://www.es.net/assets/pubs_presos/High-throughput-lessons-from-the-LHC-experience.Johnston.TNC2013.pdf
- Use Case 40 Belle II High Energy Physics Experiment <http://belle2.kek.jp/>
- Use Case 41 EISCAT 3D incoherent scatter radar system <https://www.eiscat3d.se/>
- Use Case 42 ENVRI, Common Operations of Environmental Research Infrastructure, ENVRI Project website, ENVRI Reference Model, ENVRI deliverable D3.2 : Analysis of common requirements of Environmental Research Infrastructures, ICOS, Euro-Argo, EISCAT 3D, LifeWatch, EPOS, EMSO
- Use Case 43 Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets <https://www.cresis.ku.edu/>
- Use Case 44 UAVSAR Data Processing, Data Product Delivery, and Data Services <http://uavsar.jpl.nasa.gov/>, <http://www.asf.alaska.edu/program/sdc>, <http://geo-gateway.org/main.html>
- Use Case 47 Atmospheric Turbulence - Event Discovery and Predictive Analytics <http://oceanworld.tamu.edu/resources/oceanography-book/teleconnections.htm>, <http://www.forbes.com/sites/toddwoody/2012/03/21/meet-the-scientists-mining-big-data-to-predict-the-weather/>
- Use Case 48 Climate Studies using the Community Earth System Model at DOE's NERSC center <http://www-pcmdi.llnl.gov/>, <http://www.nersc.gov/>, <http://science.energy.gov/ber/research/cesd/>, <http://www2.cisl.ucar.edu/>
- Use Case 50 DOE-BER AmeriFlux and FLUXNET Networks <http://ameriflux.lbl.gov/>, <http://www.fluxdata.org/default.aspx>
- Use Case 51 Consumption forecasting in Smart Grids <http://smartgrid.usc.edu/>, http://ganges.usc.edu/wiki/Smart_Grid, https://www.ladwp.com/ladwp/faces/ladwp/aboutus/a-power/a-p-smartgridla?_afLoop=157401916661989&_afWindowMode=0&_afWindowId=null#%40%3F_afWindowId%3Dnull%26_afLoop%3D157401916661989%26_afWindowMode%3D0%26_adf.ctrl-state%3Db7yulr4rl_17, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6475927>



63. Health Informatics

Health Informatics (131) 

This section starts by discussing general aspects of Big Data and Health including data sizes, different areas including genomics, EBI, radiology and the Quantified Self movement. We review current state of health care and trends associated with it including increased use of Telemedicine. We summarize an industry survey by GE and Accenture and an impressive exemplar Cloud-based medicine system from Potsdam. We give some details of big data in medicine. Some remarks on Cloud computing and Health focus on security and privacy issues.

We survey an April 2013 McKinsey report on the Big Data revolution in US health care; a Microsoft report in this area and a European Union report on how Big Data will allow patient centered care in the future. Examples are given of the Internet of Things, which will have great impact on health including wearables. A study looks at 4 scenarios for healthcare in 2032. Two are positive, one middle of the road and one negative. The final topic is Genomics, Proteomics and Information Visualization.

63.1 Big Data and Health

This lesson starts with general aspects of Big Data and Health including listing subareas where Big data important. Data sizes are given in radiology, genomics, personalized medicine, and the Quantified Self movement, with sizes and access to European Bioinformatics Institute.

Big Data and Health (10:02) 

<section/bigdata/applications/health.tex>

63.2 Status of Healthcare Today

This covers trends of costs and type of healthcare with low cost genomes and an aging population. Social media and government Brain initiative.

Status of Healthcare Today (16:09) 

63.3 Telemedicine (Virtual Health)

This describes increasing use of telemedicine and how we tried and failed to do this in 1994.

Telemedicine (8:21) 

63.4 Medical Big Data in the Clouds

An impressive exemplar Cloud-based medicine system from Potsdam.

Medical Big Data in the Clouds (15:02) 

63.4.1 Medical image Big Data

Medical Image Big Data (6:33) 

63.4.2 Clouds and Health

Clouds and Health (4:35) 

63.4.3 McKinsey Report on the big-data revolution in US health care

This lesson covers 9 aspects of the McKinsey report. These are the convergence of multiple positive changes has created a tipping point for innovation; Primary data pools are at the heart of the big data revolution in healthcare; Big data is changing the paradigm: these are the value pathways; Applying early successes at scale could reduce US healthcare costs by \$300 billion to \$450 billion; Most new big-data applications target consumers and providers across pathways; Innovations are weighted towards influencing individual decision-making levers; Big data innovations use a range of public, acquired, and proprietary data types; Organizations implementing a big data transformation should provide the leadership required for the associated cultural transformation; Companies must develop a range of big data capabilities.

McKinsey Report (14:53) 

63.4.4 Microsoft Report on Big Data in Health

This lesson identifies data sources as Clinical Data, Pharma & Life Science Data, Patient & Consumer Data, Claims & Cost Data and Correlational Data. Three approaches are Live data feed, Advanced analytics and Social analytics.

Microsoft Report on Big Data in Health (2:26) 

<section/bigdata/applications/health.tex>

63.4.5 EU Report on Redesigning health in Europe for 2020

This lesson summarizes an EU Report on Redesigning health in Europe for 2020. The power of data is seen as a lever for change in My Data, My decisions; Liberate the data; Connect up everything; Revolutionize health; and Include Everyone removing the current correlation between health and wealth.

EU Report on Redesigning health in Europe for 2020 (5:00) 

63.4.6 Medicine and the Internet of Things

The Internet of Things will have great impact on health including telemedicine and wearables. Examples are given.

Medicine and the Internet of Things (8:17) 

63.4.7 Extrapolating to 2032

A study looks at 4 scenarios for healthcare in 2032. Two are positive, one middle of the road and one negative.

Extrapolating to 2032 (15:13) 

63.4.8 Genomics, Proteomics and Information Visualization

A study of an Azure application with an Excel frontend and a cloud BLAST backend starts this lesson. This is followed by a big data analysis of personal genomics and an analysis of a typical DNA sequencing analytics pipeline. The Protein Sequence Universe is defined and used to motivate Multi dimensional Scaling MDS. Sammon's method is defined and its use illustrated by a metagenomics example. Subtleties in use of MDS include a monotonic mapping of the dissimilarity function. The application to the COG Proteomics dataset is discussed. We note that the MDS approach is related to the well known chisq method and some aspects of nonlinear minimization of chisq (Least Squares) are discussed.

Genomics, Proteomics and Information Visualization (6:56) 

Next we continue the discussion of the COG Protein Universe introduced in the last lesson. It is shown how Proteomics clusters are clearly seen in the Universe browser. This motivates a side remark on different clustering methods applied to metagenomics. Then we discuss the Generative Topographic Map GTM method that can be used in dimension reduction when original data is in a metric space and is in this case faster than MDS as GTM computational complexity scales like N not N squared as seen in MDS.

Examples are given of GTM including an application to topic models in Information Retrieval. Indiana University has developed a deterministic annealing improvement of GTM. 3 separate clusterings are projected for visualization and show very different structure emphasizing the importance of visualizing results of data analytics. The final slide shows an application of MDS to generate and visualize phylogenetic trees.

TODO: These two videos need to be uploaded to youtube

Genomics, Proteomics and Information Visualization I (10:33) 

Genomics, Proteomics and Information Visualization: II (7:41) 

Proteomics and Information Visualization (131) 

63.4.9 Resources

TODO: These resources have not all been checked to see if they still exist this is currently in progress

- <https://wiki.nci.nih.gov/display/CIP/CIP+Survey+of+Biomedical+Imaging+Archives>
- <http://grids.ucs.indiana.edu/ptliupages/publications/Where%20does%20all%20the%20data%20come%20from%20v7.pdf>
- <http://www.ieee-icsc.org/ICSC2010/Tony%20Hey%20-%2020100923.pdf>
- <http://quantifiedself.com/larry-smarr/>
- <http://www.ebi.ac.uk/Information/Brochures/>
- <http://www.kpcb.com/internet-trends>
- <http://www.slideshare.net/drsteventucker/wearable-health-fitness-trackers-and-the-quantified-self>
- <http://www.siam.org/meetings/sdm13/sun.pdf>
- http://en.wikipedia.org/wiki/Calico_%28company%29
- http://www.slideshare.net/GSW_Worldwide/2015-health-trends
- <http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Industrial-Internet-Changing-Competitive-Landscape-Industries.pdf>
- <http://www.slideshare.net/schappy/how-realtime-analysis-turns-big-medical-data-into-precision-medicine>
- <http://medcitynews.com/2013/03/the-body-in-bytes-medical-images-as-a-source-of-healthcare-big-data-infographic/>
- http://healthinformatics.wikispaces.com/file/view/cloud_computing.ppt
- https://www.mckinsey.com/~media/mckinsey/industries/healthcare%20systems%20and%20services/our%20insights/the%20big%20data%20revolution%20in%20us%20health%20care/the_big_data_revolution_in_healthcare.ashx
- https://ec.europa.eu/eip/ageing/file/353/download_en?token=8gECi1R0
- <http://www.kpcb.com/internet-trends>
- <http://debategraph.org/Poster.aspx?aID=77>
- <http://www.delsall.org>
- http://salsahpc.indiana.edu/millionseq/mina/16SrRNA_index.html
- <http://www.geatbx.com/docu/fcnindex-01.html>
- <https://wiki.nci.nih.gov/display/CIP/CIP+Survey+of+Biomedical+Imaging+Archives>
- <http://grids.ucs.indiana.edu/ptliupages/publications/Where%20does%20all%20the%20data%20come%20from%20v7.pdf>
- <http://www.ieee-icsc.org/ICSC2010/Tony%20Hey%20-%2020100923.pdf>
- <http://quantifiedself.com/larry-smarr/>
- <http://www.ebi.ac.uk/Information/Brochures/>
- <http://www.kpcb.com/internet-trends>
- <http://www.slideshare.net/drsteventucker/wearable-health-fitness-trackers-and-the-quantified-self>
- <http://www.siam.org/meetings/sdm13/sun.pdf>
- http://en.wikipedia.org/wiki/Calico_%28company%29
- http://www.slideshare.net/GSW_Worldwide/2015-health-trends
- <http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Industrial->

Internet-Changing-Competitive-Landscape-Industries.pdf

- <http://www.slideshare.net/schappy/how-realtime-analysis-turns-big-medical-data-into-precision-medicine>
- <http://medcitynews.com/2013/03/the-body-in-bytes-medical-images-as-a-source-of-healthcare-big-data-infographic/>
- http://healthinformatics.wikispaces.com/file/view/cloud_computing.ppt
- <http://www.mckinsey.com/~media/McKinsey/dotcom/Insights/Health%20care/The%20big-data%20revolution%20in%20US%20health%20care/The%20big-data%20revolution%20in%20US%20health%20care%20Accelerating%20value%20and%20innovation.ashx>
- <https://partner.microsoft.com/download/global/40193764>
- http://ec.europa.eu/information_society/activities/health/docs/policy/taskforce/redesigning_health-eu-for2020-ehf-report2012.pdf
- <http://www.kpcb.com/internet-trends>
- <http://www.liveathos.com/apparel/app>
- <http://debategraph.org/Poster.aspx?aID=77>
- <http://www.oerc.ox.ac.uk/downloads/presentations-from-events/microsoftworkshop/gannon>
- <http://www.delsall.org>
- http://salsahpc.indiana.edu/millionseq/mina/16SrRNA_index.html
- <http://www.geatbx.com/docu/fcnindex-01.html>



64. e-Commerce and LifeStyle

Recommender systems operate under the hood of such widely recognized sites as Amazon, eBay, Monster and Netflix where everything is a recommendation. This involves a symbiotic relationship between vendor and buyer whereby the buyer provides the vendor with information about their preferences, while the vendor then offers recommendations tailored to match their needs. Kaggle competitions improve the success of the Netflix and other recommender systems. Attention is paid to models that are used to compare how changes to the systems affect their overall performance. It is interesting that the humble ranking has become such a dominant driver of the world's economy. More examples of recommender systems are given from Google News, Retail stores and in depth Yahoo! covering the multi-faceted criteria used in deciding recommendations on web sites.

The formulation of recommendations in terms of points in a space or bag is given where bags of item properties, user properties, rankings and users are useful. Detail is given on basic principles behind recommender systems: user-based collaborative filtering, which uses similarities in user rankings to predict their interests, and the Pearson correlation, used to statistically quantify correlations between users viewed as points in a space of items. Items are viewed as points in a space of users in item-based collaborative filtering. The Cosine Similarity is introduced, the difference between implicit and explicit ratings and the k Nearest Neighbors algorithm. General features like the curse of dimensionality in high dimensions are discussed. A simple Python k Nearest Neighbor code and its application to an artificial data set in 3 dimensions is given. Results are visualized in Matplotlib in 2D and with Plotviz in 3D. The concept of a training and a testing set are introduced with training set pre labeled. Recommender systems are used to discuss clustering with k-means based clustering methods used and their results examined in Plotviz. The original labelling is compared to clustering results and extension to 28 clusters given. General issues in clustering are discussed including local optima, the use of annealing to avoid this and value of heuristic algorithms.

64.1 Recommender Systems

We introduce Recommender systems as an optimization technology used in a variety of applications and contexts online. They operate in the background of such widely recognized sites as Amazon, eBay, Monster and Netflix where everything is a recommendation. This involves a symbiotic relationship between vendor and buyer whereby the buyer provides the vendor with information about their preferences, while the vendor then offers recommendations tailored to match their needs, to the benefit of both.

There follows an exploration of the Kaggle competition site, other recommender systems and Netflix, as well as competitions held to improve the success of the Netflix recommender system. Finally attention is paid to models that are used to compare how changes to the systems affect their overall performance. It is interesting how the humble ranking has become such a dominant driver of the world's economy.

[45 \(Recommender\)](#)  PDF

64.1.1 Recommender Systems as an Optimization Problem

We define a set of general recommender systems as matching of items to people or perhaps collections of items to collections of people where items can be other people, products in a store, movies, jobs, events, web pages etc. We present this as ‘yet another optimization problem’.

[Recommender Systems I \(8:06\)](#) 

64.1.2 Recommender Systems Introduction

We give a general discussion of recommender systems and point out that they are particularly valuable in long tail of items (to be recommended) that aren't commonly known. We pose them as a rating system and relate them to information retrieval rating systems. We can contrast recommender systems based on user profile and context; the most familiar collaborative filtering of others ranking; item properties; knowledge and hybrid cases mixing some or all of these.

[Recommender Systems Introduction \(12:56\)](#) 

64.1.3 Kaggle Competitions

We look at Kaggle competitions with examples from web site. In particular we discuss an Irvine class project involving ranking jokes.

[Kaggle Competitions: \(3:36\)](#) 

Indiana University

Please note that we do not accept any projects using kaggle data for this class.

64.1.4 Examples of Recommender Systems

We go through a list of 9 recommender systems from the same Irvine class.

[Examples of Recommender Systems \(1:00\)](#) 

[section/bigdata/applications/lifestyle.tex](#)

64.1.5 Netflix on Recommender Systems

We summarize some interesting points from a tutorial from Netflix for whom *everything is a recommendation*. Rankings are given in multiple categories and categories that reflect user interests are especially important. Criteria used include explicit user preferences, implicit based on ratings and hybrid methods as well as freshness and diversity. Netflix tries to explain the rationale of its recommendations. We give some data on Netflix operations and some methods used in its recommender systems. We describe the famous Netflix Kaggle competition to improve its rating system. The analogy to maximizing click through rate is given and the objectives of optimization are given.

[Netflix on Recommender Systems \(14:20\)](#) 

Next we go through Netflix's methodology in letting data speak for itself in optimizing the recommender engine. An example is given on choosing self produced movies. A/B testing is discussed with examples showing how testing does allow optimizing of sophisticated criteria. This lesson is concluded by comments on Netflix technology and the full spectrum of issues that are involved including user interface, data, AB testing, systems and architectures. We comment on optimizing for a household rather than optimizing for individuals in household.

[Consumer Data Science \(13:04\)](#) 

64.1.6 Other Examples of Recommender Systems

We continue the discussion of recommender systems and their use in e-commerce. More examples are given from Google News, Retail stores and in depth Yahoo! covering the multi-faceted criteria used in deciding recommendations on web sites. Then the formulation of recommendations in terms of points in a space or bag is given.

Here bags of item properties, user properties, rankings and users are useful. Then we go into detail on basic principles behind recommender systems: user-based collaborative filtering, which uses similarities in user rankings to predict their interests, and the Pearson correlation, used to statistically quantify correlations between users viewed as points in a space of items.

[49 \(Recommender\)](#)  PDF

We start with a quick recap of recommender systems from previous unit; what they are with brief examples.

[Recap and Examples of Recommender Systems \(5:48\)](#) 

Examples of Recommender Systems

We give 2 examples in more detail: namely Google News and Markdown in Retail.

[Examples of Recommender Systems \(8:34\)](#) 

Recommender Systems in Yahoo Use Case Example

We describe in greatest detail the methods used to optimize Yahoo web sites. There are two lessons discussing general approach and a third lesson examines a particular personalized Yahoo page with its different components. We point out the different criteria that must be blended in making decisions; these criteria include analysis of what user does after a particular page is clicked; is the user satisfied and cannot that we quantified by purchase decisions etc. We need to choose

Articles, ads, modules, movies, users, updates, etc to optimize metrics such as relevance score, CTR, revenue, engagement. These lessons stress that if though we have big data, the recommender data is sparse. We discuss the approach that involves both batch (offline) and on-line (real time) components.

Recap of Recommender Systems II (8:46) 

Recap of Recommender Systems III (10:48) 

Case Study of Recommender systems (3:21) 

User-based nearest-neighbor collaborative filtering

Collaborative filtering is a core approach to recommender systems. There is user-based and item-based collaborative filtering and here we discuss the user-based case. Here similarities in user rankings allow one to predict their interests, and typically this is quantified by the Pearson correlation, used to statistically quantify correlations between users.

User-based nearest-neighbor collaborative filtering I (7:20) 

User-based nearest-neighbor collaborative filtering II (7:29) 

Vector Space Formulation of Recommender Systems

We go through recommender systems thinking of them as formulated in a funny vector space. This suggests using clustering to make recommendations.

Vector Space Formulation of Recommender Systems new (9:06) 

64.1.7 Resources

- <http://pages.cs.wisc.edu/~bee chung/icml11-tutorial/>

64.2 Item-based Collaborative Filtering and its Technologies

We move on to item-based collaborative filtering where items are viewed as points in a space of users. The Cosine Similarity is introduced, the difference between implicit and explicit ratings and the k Nearest Neighbors algorithm. General features like the curse of dimensionality in high dimensions are discussed.

18 (Filtering)  PDF

64.2.1 Item-based Collaborative Filtering

We covered user-based collaborative filtering in the previous unit. Here we start by discussing memory-based real time and model based offline (batch) approaches. Now we look at item-based collaborative filtering where items are viewed in the space of users and the cosine measure is used to quantify distances. We discuss optimizations and how batch processing can help. We discuss different Likert ranking scales and issues with new items that do not have a significant number of rankings.

Item Based Filtering (11:18) 

section/bigdata/applications/lifestyle.tex

k Nearest Neighbors and High Dimensional Spaces (7:16) 

64.2.2 k-Nearest Neighbors and High Dimensional Spaces

We define the k Nearest Neighbor algorithms and present the Python software but do not use it. We give examples from Wikipedia and describe performance issues. This algorithm illustrates the curse of dimensionality. If items were a real vectors in a low dimension space, there would be faster solution methods.

k Nearest Neighbors and High Dimensional Spaces (10:03) 

64.2.3 Resources

TODO: These resources have not all been checked to see if they still exist this is currently in progress

- <http://www.slideshare.net/xamat/building-largescale-realworld-recommender-systems-recsys2012-tutorial>
- http://www.ifi.uzh.ch/ce/teaching/spring2012/16-Recommender-Systems_Slides.pdf
- <https://www.kaggle.com/>
- http://www.ics.uci.edu/~welling/teaching/CS77Bwinter12/CS77B_w12.html
- Jeff Hammerbacher <https://berkeleydatascience.files.wordpress.com/2012/01/20120117berkeley1.pdf>
- <http://www.techworld.com/news/apps/netflix-foretells-house-of-cards-success-with-cassandra-big-data-engine-3437514/>
- https://en.wikipedia.org/wiki/A/B_testing
- <http://www.infoq.com/presentations/Netflix-Architecture>




65. Physics

This section starts by describing the LHC accelerator at CERN and evidence found by the experiments suggesting existence of a Higgs Boson. The huge number of authors on a paper, remarks on histograms and Feynman diagrams is followed by an accelerator picture gallery. The next unit is devoted to Python experiments looking at histograms of Higgs Boson production with various forms of shape of signal and various background and with various event totals. Then random variables and some simple principles of statistics are introduced with explanation as to why they are relevant to Physics counting experiments. The unit introduces Gaussian (normal) distributions and explains why they seen so often in natural phenomena. Several Python illustrations are given. Random Numbers with their Generators and Seeds lead to a discussion of Binomial and Poisson Distribution. Monte-Carlo and accept-reject methods. The Central Limit Theorem concludes discussion.

65.1 Looking for Higgs Particles

65.1.1 Bumps in Histograms, Experiments and Accelerators

This unit is devoted to Python and Java experiments looking at histograms of Higgs Boson production with various forms of shape of signal and various background and with various event totals. The lectures use Python but use of Java is described.

Higgs (20) 

HiggsClassI-Sloping.py 

section/bigdata/applications/physics.tex

65.1.2 Particle Counting

We return to particle case with slides used in introduction and stress that particles often manifested as bumps in histograms and those bumps need to be large enough to stand out from background in a statistically significant fashion.

Discovery of Higgs Particle (13:49) 

We give a few details on one LHC experiment ATLAS. Experimental physics papers have a staggering number of authors and quite big budgets. Feynman diagrams describe processes in a fundamental fashion.

Looking for Higgs Particle and Counting Introduction II (7:38) 


65.1.3 Experimental Facilities

We give a few details on one LHC experiment ATLAS. Experimental physics papers have a staggering number of authors and quite big budgets. Feynman diagrams describe processes in a fundamental fashion.

Looking for Higgs Particle Experiments (9:29) 

65.1.4 Accelerator Picture Gallery of Big Science

This lesson gives a small picture gallery of accelerators. Accelerators, detection chambers and magnets in tunnels and a large underground laboratory used for experiments where you need to be shielded from background like cosmic rays.

Accelerator Picture Gallery of Big Science (11:21) 

65.1.5 Resources

- <http://grids.ucs.indiana.edu/ptliupages/publications/Where%20does%20all%20the%20data%20come%20from%20v7.pdf>
- <http://www.sciencedirect.com/science/article/pii/S037026931200857X>
- <http://www.nature.com/news/specials/lhc/interactive.html>

Looking for Higgs Particles: Python Event Counting for Signal and Background (Part 2)

This unit is devoted to Python experiments looking at histograms of Higgs Boson production with various forms of shape of signal and various background and with various event totals.

Higgs II (29)  PDF

Files:

HiggsClassI-Sloping.py 

HiggsClassIII.py 

HiggsClassIIUniform.py 

section/bigdata/applications/physics.tex

65.1.6 Event Counting

We define *event counting* data collection environments. We discuss the python and Java code to generate events according to a particular scenario (the important idea of Monte Carlo data). Here a sloping background plus either a Higgs particle generated similarly to LHC observation or one observed with better resolution (smaller measurement error).

Event Counting (7:02) 

65.1.7 Monte Carlo

This uses Monte Carlo data both to generate data like the experimental observations and explore effect of changing amount of data and changing measurement resolution for Higgs.

With Python examples of Signal plus Background (7:33) 

This lesson continues the examination of Monte Carlo data looking at effect of change in number of Higgs particles produced and in change in shape of background.

Change shape of background & num of Higgs Particles (7:01) 

65.1.8 Resources

- Python for Data Analysis: Agile Tools for Real World Data By Wes McKinney, Publisher: O'Reilly Media, Released: October 2012, Pages: 472.
- <http://jwork.org/scavis/api/>
- <https://en.wikipedia.org/wiki/DataMelt>

65.1.9 Random Variables, Physics and Normal Distributions

We introduce random variables and some simple principles of statistics and explains why they are relevant to Physics counting experiments. The unit introduces Gaussian (normal) distributions and explains why they seen so often in natural phenomena. Several Python illustrations are given. Java is currently not available in this unit.

Higgs (39) 

HiggsClassIII.py 

65.1.10 Statistics Overview and Fundamental Idea: Random Variables

We go through the many different areas of statistics covered in the Physics unit. We define the statistics concept of a random variable.

Random variables and normal distributions (8:19) 

65.1.11 Physics and Random Variables

We describe the DIKW pipeline for the analysis of this type of physics experiment and go through details of analysis pipeline for the LHC ATLAS experiment. We give examples of event displays showing the final state particles seen in a few events. We illustrate how physicists decide whats going on with a plot of expected Higgs production experimental cross sections (probabilities) for

section/bigdata/applications/physics.tex

signal and background.

Physics and Random Variables I (8:34) 

Physics and Random Variables II (5:50) 

65.1.12 Statistics of Events with Normal Distributions

We introduce Poisson and Binomial distributions and define independent identically distributed (IID) random variables. We give the law of large numbers defining the errors in counting and leading to Gaussian distributions for many things. We demonstrate this in Python experiments.

Statistics of Events with Normal Distributions (11:25) 

65.1.13 Gaussian Distributions

We introduce the Gaussian distribution and give Python examples of the fluctuations in counting Gaussian distributions.

Gaussian Distributions (9:08) 

65.1.14 Using Statistics

We discuss the significance of a standard deviation and role of biases and insufficient statistics with a Python example in getting incorrect answers.

Using Statistics (14:02) 

65.1.15 Resources

- <http://indico.cern.ch/event/20453/session/6/contribution/15?materialId=slides>
- <http://www.atlas.ch/photos/events.html>
- <https://cms.cern/>

65.1.16 Random Numbers, Distributions and Central Limit Theorem

We discuss Random Numbers with their Generators and Seeds. It introduces Binomial and Poisson Distribution. Monte-Carlo and accept-reject methods are discussed. The Central Limit Theorem and Bayes law concludes discussion. Python and Java (for student - not reviewed in class) examples and Physics applications are given.

Higgs III (44)  PDF

Files:

HiggsClassIII.py 

Generators and Seeds

We define random numbers and describe how to generate them on the computer giving Python examples. We define the seed used to define to specify how to start generation.

section/bigdata/applications/physics.tex

[Higgs Particle Counting Errors \(6:28\)](#) 

[Generators and Seeds II \(7:10\)](#) 

Binomial Distribution

We define binomial distribution and give LHC data as an example of where this distribution valid.

[Binomial Distribution: \(12:38\)](#) 

Accept-Reject

We introduce an advanced method **accept/reject** for generating random variables with arbitrary distributions.

[Accept-Reject \(5:54\)](#) 

Monte Carlo Method

We define Monte Carlo method which usually uses accept/reject method in typical case for distribution.

[Monte Carlo Method \(2:23\)](#) 

Poisson Distribution

We extend the Binomial to the Poisson distribution and give a set of amusing examples from Wikipedia.

[Poisson Distribution \(4:37\)](#) 

Central Limit Theorem

We introduce Central Limit Theorem and give examples from Wikipedia.

[Central Limit Theorem \(4:47\)](#) 

Interpretation of Probability: Bayes v. Frequency

This lesson describes difference between Bayes and frequency views of probability. Bayes's law of conditional probability is derived and applied to Higgs example to enable information about Higgs from multiple channels and multiple experiments to be accumulated.

[Interpretation of Probability \(12:39\)](#) 

Resources

TODO: integrate physics-references.bib

65.2 SKA – Square Kilometer Array

Professor Diamond, accompanied by Dr. Rosie Bolton from the SKA Regional Centre Project gave a presentation at SC17 ‘‘into the deepest reaches of the observable universe as they describe the

[section/bigdata/applications/physics.tex](#)

SKA’s international partnership that will map and study the entire sky in greater detail than ever before.’’

- <http://sc17.supercomputing.org/presentation/?id=inspkr101&sess=sess263>

A summary article about this effort is available at:

- <https://www.hpcwire.com/2017/11/17/sc17-keynote-hpc-powers-ska-efforts-peer-deep-cosmos/>


The video is hosted at

- <http://sc17.supercomputing.org/presentation/?id=inspkr101&sess=sess263>

Start at about 1:03:00 (e.g. the one hour mark)

65.3 Radar

The changing global climate is suspected to have long-term effects on much of the world’s inhabitants. Among the various effects, the rising sea level will directly affect many people living in low-lying coastal regions. While the ocean’s thermal expansion has been the dominant contributor to rises in sea level, the potential contribution of discharges from the polar ice sheets in Greenland and Antarctica may provide a more significant threat due to the unpredictable response to the changing climate. The Radar-Informatics unit provides a glimpse in the processes fueling global climate change and explains what methods are used for ice data acquisitions and analysis.

Radar (58) 

65.3.1 Introduction

This lesson motivates radar-informatics by building on previous discussions on why X-applications are growing in data size and why analytics are necessary for acquiring knowledge from large data. The lesson details three mosaics of a changing Greenland ice sheet and provides a concise overview to subsequent lessons by detailing explaining how other remote sensing technologies, such as the radar, can be used to sound the polar ice sheets and what we are doing with radar images to extract knowledge to be incorporated into numerical models.

Radar Informatics (3:31) 

65.3.2 Remote Sensing

This lesson explains the basics of remote sensing, the characteristics of remote sensors and remote sensing applications. Emphasis is on image acquisition and data collection in the electromagnetic spectrum.

Remote Sensing (6:43) 

65.3.3 Ice Sheet Science

This lesson provides a brief understanding on why melt water at the base of the ice sheet can be detrimental and why it’s important for sensors to sound the bedrock.

Ice Sheet Science (1:00) 

65.3.4 Global Climate Change

This lesson provides an understanding and the processes for the greenhouse effect, how warming effects the Polar Regions, and the implications of a rise in sea level.

Global Climate Change (2:51) 

65.3.5 Radio Overview

This lesson provides an elementary introduction to radar and its importance to remote sensing, especially to acquiring information about Greenland and Antarctica.

Radio Overview (4:16) 

65.3.6 Radio Informatics


This lesson focuses on the use of sophisticated computer vision algorithms, such as active contours and a hidden markov model to support data analysis for extracting layers, so ice sheet models can accurately forecast future changes in climate.


Radio Informatics (3:35) 



66. Sensors

We start with the Internet of Things IoT giving examples like monitors of machine operation, QR codes, surveillance cameras, scientific sensors, drones and self driving cars and more generally transportation systems. We give examples of robots and drones. We introduce the Industrial Internet of Things IIoT and summarize surveys and expectations Industry wide. We give examples from General Electric. Sensor clouds control the many small distributed devices of IoT and IIoT. More detail is given for radar data gathered by sensors; ubiquitous or smart cities and homes including U-Korea; and finally the smart electric grid.

Sensor I (31) 

Sensor II (44) 

66.1 Internet of Things

There are predicted to be 24-50 Billion devices on the Internet by 2020; these are typically some sort of sensor defined as any source or sink of time series data. Sensors include smartphones, webcams, monitors of machine operation, barcodes, surveillance cameras, scientific sensors (especially in earth and environmental science), drones and self driving cars and more generally transportation systems. The lesson gives many examples of distributed sensors, which form a Grid that is controlled by a cloud.

Internet of Things (12:36) 

66.2 Robotics and IoT

Examples of Robots and Drones.

<section/bigdata/applications/sensor.tex>

Robotics and IoT Expectations (8:05) 

66.3 Industrial Internet of Things

We summarize surveys and expectations Industry wide.

Industrial Internet of Things (24:02) 

66.4 Sensor Clouds

We describe the architecture of a Sensor Cloud control environment and gives example of interface to an older version of it. The performance of system is measured in terms of processing latency as a function of number of involved sensors with each delivering data at 1.8 Mbps rate.

Sensor Clouds (4:40) 

66.5 Earth/Environment/Polar Science data gathered by Sensors

This lesson gives examples of some sensors in the Earth/Environment/Polar Science field. It starts with material from the CReSIS polar remote sensing project and then looks at the NSF Ocean Observing Initiative and NASA's MODIS or Moderate Resolution Imaging Spectroradiometer instrument on a satellite.

Earth/Environment/Polar Science data gathered by Sensors (4:58) 

66.6 Ubiquitous/Smart Cities

For Ubiquitous/Smart cities we give two examples: Iniquitous Korea and smart electrical grids.

Ubiquitous/Smart Cities (1:44) 

66.7 U-Korea (U=Ubiquitous)

Korea has an interesting positioning where it is first worldwide in broadband access per capita, e-government, scientific literacy and total working hours. However it is far down in measures like quality of life and GDP. U-Korea aims to improve the latter by Pervasive computing, everywhere, anytime i.e. by spreading sensors everywhere. The example of a 'High-Tech Utopia' New Songdo is given.

U-Korea (U=Ubiquitous) (2:49) 

66.8 Smart Grid

The electrical Smart Grid aims to enhance USA's aging electrical infrastructure by pervasive deployment of sensors and the integration of their measurement in a cloud or equivalent server infrastructure. A variety of new instruments include smart meters, power monitors, and measures

section/bigdata/applications/sensor.tex

of solar irradiance, wind speed, and temperature. One goal is autonomous local power units where good use is made of waste heat.

Smart Grid (6:04) 

66.9 Resources

TODO: These resources have not all been checked to see if they still exist this is currently in progress

- <https://www.gesoftware.com/minds-and-machines>
- <https://www.gesoftware.com/predix>
- <https://www.gesoftware.com/sites/default/files/the-industrial-internet/index.html>
- <https://developer.cisco.com/site/eiot/discover/overview/>
- <http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Industrial-Internet-Changing-Competitive-Landscape-Industries.pdf>
- <http://www.gesoftware.com/ge-predictivity-infographic>
- <http://www.getransportation.com/railconnect360/rail-landscape>
- <http://www.gesoftware.com/sites/default/files/GE-Software-Modernizing-Machine-to-Machine-Interactions.pdf>



67. Sports

Sports sees significant growth in analytics with pervasive statistics shifting to more sophisticated measures. We start with baseball as game is built around segments dominated by individuals where detailed (video/image) achievement measures including PITCHf/x and FIELDf/x are moving field into big data arena. There are interesting relationships between the economics of sports and big data analytics. We look at Wearables and consumer sports/recreation. The importance of spatial visualization is discussed. We look at other Sports: Soccer, Olympics, NFL Football, Basketball, Tennis and Horse Racing.

67.1 Basic Sabermetrics

This unit discusses baseball starting with the movie Moneyball and the 2002-2003 Oakland Athletics. Unlike sports like basketball and soccer, most baseball action is built around individuals often interacting in pairs. This is much easier to quantify than many player phenomena in other sports. We discuss Performance-Dollar relationship including new stadiums and media/advertising. We look at classic baseball averages and sophisticated measures like Wins Above Replacement.

[Overview \(40\)](#) 

67.1.1 Introduction and Sabermetrics (Baseball Informatics) Lesson

Introduction to all Sports Informatics, Moneyball The 2002-2003 Oakland Athletics, Diamond Dollars economic model of baseball, Performance - Dollar relationship, Value of a Win.

[Introduction and Sabermetrics \(Baseball Informatics\) Lesson \(31:4\)](#) 

<section/bigdata/applications/sport.tex>

67.1.2 Basic Sabermetrics

Different Types of Baseball Data, Sabermetrics, Overview of all data, Details of some statistics based on basic data, OPS, wOBA, ERA, ERC, FIP, UZR.

Basic Sabermetrics (26:53) 


67.1.3 Wins Above Replacement

Wins above Replacement WAR, Discussion of Calculation, Examples, Comparisons of different methods, Coefficient of Determination, Another, Sabermetrics Example, Summary of Sabermetrics.

Wins Above Replacement (30:43) 

67.2 Advanced Sabermetrics

This unit discusses ‘advanced sabermetrics’ covering advances possible from using video from PITCHf/X, FIELDf/X, HITf/X, COMMANDf/X and MLBAM.

Sporta II (41) 

67.2.1 Pitching Clustering

A Big Data Pitcher Clustering method introduced by Vince Gennaro, Data from Blog and video at 2013 SABR conference.

Pitching Clustering (20:59) 


67.2.2 Pitcher Quality

Results of optimizing match ups, Data from video at 2013 SABR conference.

Pitcher Quality (10:02) 

67.3 PITCHf/X

Examples of use of PITCHf/X.

PITCHf/X (10:39) 

67.3.1 Other Video Data Gathering in Baseball

FIELDf/X, MLBAM, HITf/X, COMMANDf/X.

Other Video Data Gathering in Baseball (18:5) 

67.4 Other Sports

We look at Wearables and consumer sports/recreation. The importance of spatial visualization is discussed. We look at other Sports: Soccer, Olympics, NFL Football, Basketball, Tennis and Horse

section/bigdata/applications/sport.tex

Racing.

44 (Sports III) 

67.4.1 Wearables

Consumer Sports, Stake Holders, and Multiple Factors.

Wearables (22:2) 

67.4.2 Soccer and the Olympics

Soccer, Tracking Players and Balls, Olympics.

Soccer and the Olympics (8:28) 

67.4.3 Spatial Visualization in NFL and NBA

NFL, NBA, and Spatial Visualization.

Spatial Visualization in NFL and NBA (15:19) 

67.4.4 Tennis and Horse Racing

Tennis, Horse Racing, and Continued Emphasis on Spatial Visualization.

Tennis and Horse Racing (8:52) 

67.4.5 Resources

TODO: These resources have not all been checked to see if they still exist this is currently in progress

- http://www.sloansportsconference.com/?page_id=481&sort_cate=Research%20Paper
- http://www.slideshare.net/Tricon_Infotech/big-data-for-big-sports
- <http://www.slideshare.net/BrandEmotivity/sports-analytics-innovation-summit-data-powered-storytelling>
- <http://www.liveathos.com/apparel/app>
- <http://www.slideshare.net/elew/sport-analytics-innovation>
- <http://www.wired.com/2013/02/catapult-smartball/>
- http://www.sloansportsconference.com/wp-content/uploads/2014/06/Automated_Playbook_Generation.pdf
- <http://autoscout.adsc.illinois.edu/publications/football-trajectory-dataset/>
- http://www.sloansportsconference.com/wp-content/uploads/2012/02/Goldsberry_Sloan_Submission.pdf
- <http://gamesetmap.com/>
- <http://www.trakus.com/technology.asp#tNetText>
- <http://www.slideshare.net/BrandEmotivity/sports-analytics-innovation-summit-data-powered-storytelling>
- <http://www.sloansportsconference.com/>

- <http://sabr.org/>
- <http://en.wikipedia.org/wiki/Sabermetrics>
- http://en.wikipedia.org/wiki/Baseball_statistics
- <http://www.sportvision.com/baseball>
- <http://m.mlb.com/news/article/68514514/mlbam-introduces-new-way-to-analyze-every-play>
- <http://www.fangraphs.com/library/offense/offensive-statistics-list/>
- http://en.wikipedia.org/wiki/Component_ERA
- <http://www.fangraphs.com/library/pitching/fip/>
- <http://nomaas.org/2012/05/a-look-at-the-defense-the-yankees-d-stinks-edition/>
- http://en.wikipedia.org/wiki/Wins_Above_Replacement
- <http://www.fangraphs.com/library/misc/war/>
- http://www.baseball-reference.com/about/war_explained.shtml
- http://www.baseball-reference.com/about/war_explained_comparison.shtml
- http://www.baseball-reference.com/about/war_explained_position.shtml
- http://www.baseball-reference.com/about/war_explained_pitch.shtml
- <http://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=y&type=8&season=2014&month=0&season1=1871&ind=0>
- <http://battingleadoff.com/2014/01/08/comparing-the-three-war-measures-part-ii/>
- <http://battingleadoff.com/2014/01/08/comparing-the-three-war-measures-part-ii/>
- http://en.wikipedia.org/wiki/Coefficient_of_determination
- http://www.sloansportsconference.com/wp-content/uploads/2014/02/2014_SSAC_Data-driven-Method-for-In-game-Decision-Making.pdf
- <https://courses.edx.org/courses/BUx/SABR101x/2T2014/courseware/10e616fc7649469ab4457ae>
- <http://vincegennaro.mlblogs.com/>
- https://www.youtube.com/watch?v=H-kx-x_dOMk
- <http://www.sportvision.com/media/pitchfx-how-it-works>
- <http://www.baseballprospectus.com/article.php?articleid=13109>
- <http://baseball.physics.illinois.edu/FastPFXGuide.pdf>
- <http://baseball.physics.illinois.edu/FieldFX-TDR-GregR.pdf>
- <http://www.sportvision.com/baseball/fieldfx>
- <http://regressing.deadspin.com/mlb-announces-revolutionary-new-fielding-tracking-system-1534200504>
- <http://grantland.com/the-triangle/mlb-advanced-media-play-tracking-bob-bowman-interview/>
- <http://www.sportvision.com/baseball/hitfx>
- <https://www.youtube.com/watch?v=YkjtNuNmK74>



68. Web Search and Text Mining

This section starts with an overview of data mining and puts our study of classification, clustering and exploration methods in context. We examine the problem to be solved in web and text search and note the relevance of history with libraries, catalogs and concordances. An overview of web search is given describing the continued evolution of search engines and the relation to the field of Information.

The importance of recall, precision and diversity is discussed. The important Bag of Words model is introduced and both Boolean queries and the more general fuzzy indices. The important vector space model and revisiting the Cosine Similarity as a distance in this bag follows. The basic TF-IDF approach is discussed. Relevance is discussed with a probabilistic model while the distinction between Bayesian and frequency views of probability distribution completes this unit.

We start with an overview of the different steps (data analytics) in web search and then goes key steps in detail starting with document preparation. An inverted index is described and then how it is prepared for web search. The Boolean and Vector Space approach to query processing follow. This is followed by Link Structure Analysis including Hubs, Authorities and PageRank. The application of PageRank ideas as reputation outside web search is covered. The web graph structure, crawling it and issues in web advertising and search follow. The use of clustering and topic models completes the section.

68.1 Web Search and Text Mining

The unit starts with the web with its size, shape (coming from the mutual linkage of pages by URL's) and universal power laws for number of pages with particular number of URL's linking out or in to page. Information retrieval is introduced and compared to web search. A comparison is given between semantic searches as in databases and the full text search that is base of Web search. The origin of web search in libraries, catalogs and concordances is summarized. DIKW -- Data

Information Knowledge Wisdom -- model for web search is discussed. Then features of documents, collections and the important Bag of Words representation. Queries are presented in context of an Information Retrieval architecture. The method of judging quality of results including recall, precision and diversity is described. A time line for evolution of search engines is given.

Boolean and Vector Space models for query including the cosine similarity are introduced. Web Crawlers are discussed and then the steps needed to analyze data from Web and produce a set of terms. Building and accessing an inverted index is followed by the importance of term specificity and how it is captured in TF-IDF. We note how frequencies are converted into belief and relevance.

Web Search and Text Mining (56) 

68.1.1 The Problem

Text Mining (9:56) 

This lesson starts with the web with its size, shape (coming from the mutual linkage of pages by URL's) and universal power laws for number of pages with particular number of URL's linking out or in to page.

68.1.2 Information Retrieval

Information Retrieval (6:06) 

Information retrieval is introduced A comparison is given between semantic searches as in databases and the full text search that is base of Web search. The ACM classification illustrates potential complexity of ontologies. Some differences between web search and information retrieval are given.

68.1.3 History

Web Search History (5:48) 

The origin of web search in libraries, catalogs and concordances is summarized.

68.1.4 Key Fundamental Principles

Principles (9:30) 

This lesson describes the DIKW -- Data Information Knowledge Wisdom -- model for web search. Then it discusses documents, collections and the important Bag of Words representation.

68.1.5 Information Retrieval (Web Search) Components

Fundamental Principles of Web Search (5:06) 

This describes queries in context of an Information Retrieval architecture. The method of judging quality of results including recall, precision and diversity is described.

68.2 Search Engines

[Search Engines \(3:08\)](#) 

This short lesson describes a time line for evolution of search engines. The first web search approaches were directly built on Information retrieval but in 1998 the field was changed when Google was founded and showed the importance of URL structure as exemplified by PageRank.

68.2.1 Boolean and Vector Space Models

[Boolean and Vector Space Model \(6:17\)](#) 


This lesson describes the Boolean and Vector Space models for query including the cosine similarity.

68.2.2 Web crawling and Document Preparation

[Web crawling and Document Preparation \(4:55\)](#) 

This describes a Web Crawler and then the steps needed to analyze data from Web and produce a set of terms.

68.2.3 Indices

[Indices \(5:44\)](#) 


This lesson describes both building and accessing an inverted index. It describes how phrases are treated and gives details of query structure from some early logs.

68.2.4 TF-IDF and Probabilistic Models

[TF-IDF and Probabilistic Models \(3:57\)](#) 

It describes the importance of term specificity and how it is captured in TF-IDF. It notes how frequencies are converted into belief and relevance.

68.3 Topics in Web Search and Text Mining

[Text Mining \(33\)](#)  PDF

We start with an overview of the different steps (data analytics) in web search. This is followed by Link Structure Analysis including Hubs, Authorities and PageRank. The application of PageRank ideas as reputation outside web search is covered. Issues in web advertising and search follow. This leads to emerging field of computational advertising. The use of clustering and topic models completes unit with Google News as an example.

68.3.1 Data Analytics for Web Search

[Web Search and Text Mining II \(6:11\)](#) 

This short lesson describes the different steps needed in web search including: Get the digital data (from web or from scanning); Crawl web; Preprocess data to get searchable things (words,

positions); Form Inverted Index mapping words to documents; Rank relevance of documents with potentially sophisticated techniques; and integrate technology to support advertising and ways to allow or stop pages artificially enhancing relevance.

68.3.2 Link Structure Analysis including PageRank

Related Applications (17:24) 

The value of links and the concepts of Hubs and Authorities are discussed. This leads to definition of PageRank with examples. Extensions of PageRank viewed as a reputation are discussed with journal rankings and university department rankings as examples. There are many extension of these ideas which are not discussed here although topic models are covered briefly in a later lesson.

68.3.3 Web Advertising and Search

Web Advertising and Search (9:02) 

Internet and mobile advertising is growing fast and can be personalized more than for traditional media. There are several advertising types Sponsored search, Contextual ads, Display ads and different models: Cost per viewing, cost per clicking and cost per action. This leads to emerging field of computational advertising.

68.3.4 Clustering and Topic Models

Clustering and Topic Models (6:21) 

We discuss briefly approaches to defining groups of documents. We illustrate this for Google News and give an example that this can give different answers from word-based analyses. We mention some work at Indiana University on a Latent Semantic Indexing model.

68.3.5 Resources

All resources accessed March 2018.

- http://saedsayad.com/data_mining_map.htm
- http://webcourse.cs.technion.ac.il/236621/Winter2011-2012/en/ho_Lectures.html
- The Web Graph: an Overviews: <https://www.youtube.com/watch?v=yPFi6xFnDHE>
Jean-Loup Guillaume and Matthieu Latapy <https://hal.archives-ouvertes.fr/file/index/docid/54458/filename/webgraph.pdf>
- Constructing a reliable Web graph with information on browsing behavior, Yiqun Liu, Yufei Xue, Danqing Xu, Rongwei Cen, Min Zhang, Shaoping Ma, Liyun Ru <http://www.sciencedirect.com/science/article/pii/S0167923612001844>
- <http://www.ifis.cs.tu-bs.de/teaching/ss-11/irws>
- <https://en.wikipedia.org/wiki/PageRank>
- Meeker/Wu May 29 2013 Internet Trends D11 Conference <http://www.slideshare.net/kleinerperkins/kpcb-internet-trends-2013>




Cloud and Big Data Technologies


69	Big Data Applications and Software	639
69.1	Overview	
69.2	Introduction	
69.3	Application Structure	
69.4	Application Aspects	
69.5	Applications	
69.6	Other	
70	Technology Overview	643
70.1	Workflow-Orchestration	
70.2	Application and Analytics	
70.3	Application Hosting Frameworks	
70.4	High level Programming	
70.5	Streams	
70.6	Basic Programming Model and Runtime, SPMD, MapReduce	
70.7	Inter process communication Collectives	
70.8	In-memory databases/caches	
70.9	Object-relational mapping	
70.10	Extraction Tools	
70.11	SQL and SQL Services	
70.12	NoSQL	
70.13	File management	
70.14	Data Transport	
70.15	Cluster Resource Management	
70.16	File systems	
70.17	Interoperability	
70.18	DevOps	
70.19	IaaS Management from HPC to hypervisors	
70.20	Cross-Cutting Functions	
70.21	Message and Data Protocols	
70.22	Technologies To Be Integrated	
70.23	Exercise	





69. Big Data Applications and Software

69.1 Overview

Overview (Pages 26) 

Part 1 (11:29) 

Part 2 (04:10) 

Part 3 (12:41) 

69.2 Introduction

Course Introduction (Pages 39) 

Introduction (0:13:59) 

Real World Big Data (0:15:28) 

Basic Trends and Jobs (0:10:57) 

TODO: on box but not available, move to google drive

Access Patterns, Data Access Patterns and Introduction to using HPC-ABDS (Pages ???) 

A. Introduction to HPC-ABDS Software and Access Patterns (0:27:45) 

B. Science Examples (Data Access Patterns) (0:18:38) 

- Resource 1 <http://grids.ucs.indiana.edu/ptliupages/publications/HPC-ABDSDescribedv2.pdf>
- Resource 2 <http://hpc-abds.org/kaleidoscope/>

C. Remaining General Access Patterns (11:26) 

D. Summary HPC-ABDS Layers 1 - 6 (14:32) 

E. Summary HPC-ABDS Layers 7 - 13 (30:52) 

F. Summary HPC-ABDS Layers 14 - 17 (28:02) 

G. Summary HPC-ABDS Others (20:20) 

69.3 Application Structure

Big Data Application Structure, Slides <https://iu.box.com/s/z171trvqfw6vv4wnc8xr6gf1qpc9a2dr>

NIST Big Data Sub Groups (0:23:25) 

Big Data Patterns - Sources of Parallelism (0:23:51) 

First and Second Set of Features (0:18:26) 

Machine Learning Aspect of Second Feature Set and the Third Set (0:18:38) 

69.4 Application Aspects

TODO: slides are on box and not google drive Aspects of Big Data Applications, Slides <https://iu.box.com/s/atgkxucop1lzftkunf8a12fe74x65na6>

Other sources of use cases and Classical Databases/SQL Solutions (0:16:50) 

SQL Solutions - Machine Learning Example - and MapReduce (0:18:49) 

Clouds vs HPC - Data Intensive vs. Simulation Problems (0:20:26) 

69.5 Applications

TODO: slides are on box and not google drive Big Data Applications and Generalizing their Structure, Slides <https://iu.box.com/s/01dndtucmynekgehur00vktfgcpgpgc1r>

NIST UseCases and Image Based Applications Examples I (0:25:20) 

Image Based Applications II (0:15:23) 

Internet of Things Based Applications (0:25:25) 

Big Data Patterns - the Ogres and their Facets I (0:22:44) 

Facets of the Big Data Ogres II (0:15:09) 

69.6 Other

More of Software Stack (0:24:00) 



70. Technology Overview

In this section we find a number of technologies that are related to big data. Certainly a number of these projects are hosted as an Apache project. One important resource for a general list of all apache projects is at

Apache projects: <https://projects.apache.org/projects.html?category>

70.1 Workflow-Orchestration

70.1.1 ODE

Apache ODE (Orchestration Director Engine) is an open source implementation of the WS-BPEL 2.0 standard. WS-BPEL which stands for Web Services Business Process Execution Language, is an executable language for writing business processes with web services [107]. It includes control structures like conditions or loops as well as elements to invoke web services and receive messages from services. ODE uses WSDL (Web Services Description Language) for interfacing with web services [457]. Naming a few of its features, It supports two communication layers for interacting with the outside world, one based on Axis2 (Web Services http transport) and another one based on the JBI standard. It also supports both long and short living process executions for orchestrating services for applications [456].

70.1.2 ActiveBPEL

Business Process Execution Language for Web Services (BPEL4WS or just BPEL) is an XML-based grammar for describing the logic to coordinate and control web services that seamlessly integrate people, processes and systems, increasing the efficiency and visibility of the business. ActiveBPEL is a robust Java/J2EE runtime environment that is capable of executing process definitions created to the Business Process Execution Language for Web Services. The ActiveBPEL also provides an administration interface that is accessible via web service invocations;and it can

also be use to administer, to control and to integrate web services into a larger application [10].

70.1.3 Airavata

Apache Airavata [219] is a software framework that enables you to compose, manage, execute, and monitor large scale applications and workflows on distributed computing resources such as local clusters, supercomputers, computational grids, and computing clouds. Scientific gateway developers use Airavata as their middleware layer between job submissions and grid systems. Airavata supports long running applications and workflows on distributed computational resources. Many scientific gateways are already using Airavata to perform computations (e.g. Ultrascan [514], SEAGrid [323] and GenApp [218]).

-- check content --

70.1.4 Pegasus

Pegasus is workflow management system that allows to compose and execute a workflow in an application in different environment without the need for any modifications [490]. It allows users to make high level workflow without thinking about the low level details. It locates the required input data and computational resources automatically. Pegasus also maintains information about tasks done and data produced. In case of errors Pegasus tries to recover by retrying the whole workflow and providing check pointing at workflow-level. It cleans up the storage as the workflow gets executed so that data-intensive workflows can have enough required space to execute on storage-constrained resources. Some of the other advantages of Pegasus are:scalability, reliability and high performance. Pegasus has been used in many scientific domains like astronomy, bioinformatics, earthquake science, ocean science, gravitational wave physics and others.

-- check content --

70.1.5 Kepler

Kepler, scientific workflow application, is designed to help scientist, analyst, and computer programmer create, execute and share models and analyses across a broad range of scientific and engineering disciplines. Kepler can operate on data stored in a variety of formats, locally and over the internet, and is an effective environment for integrating disparate software components such as merging *R* scripts with compiled *C* code, or facilitating remote, distributed execution of models. Using Kepler's GUI, users can simply select and then connect pertinent analytical components and data sources to create a *scientific workflow*. Overall, the Kepler helps users share and reuse data, workflow, and components developed by the scientific community to address common needs [365].

70.1.6 Swift

Swift is a general-purpose, multi-paradigm, compiled programming language. It has been developed by Apple Inc. for iOS, macOS, watchOS, tvOS, and Linux. This programming language is intended to be more robust and resilient to erroneous code than Objective-C, and more concise. It has been built with the LLVM compiler framework included in Xcode 6 and later and, on platforms other than Linux. C, Objective-C, C++ and Swift code can be run within one program as Swift uses the Objective-C runtime library [739].

Swift supports the core concepts that made Objective-C flexible, notably dynamic dispatch, widespread late binding, extensible programming and similar features. Swift features have well-

known safety and performance trade-offs. A system that helps address common programming errors like null pointers was introduced to enhance safety. Apple has invested considerable effort in aggressive optimization that can flatten out method calls and accessors to eliminate this overhead to handle performance issues.

70.1.7 Taverna

Taverna is workflow management system that is transitioning to Apache Incubator as of Jan 2017 [617]. Taverna suite includes 2 products:

1. Taverna Workbench is desktop client where user can define the workflow. 2. Taverna Server is responsible for executing the remote workflows.

Taverna workflows can also be executed on command-line. Taverna supports wide range of services including WSDL-style and RESTful Web Services, BioMart, SoapLab, R, and Excel. Taverna also support mechanism to monitor the running workflows using its web browser interface. The formal syntax and operational semantics of Taverna is explained in [654].

-- check content --

70.1.8 Triana

Triana is an open source problem solving software that comes with powerful data analysis tools [652]. Having been developed at Cardiff University, it has a good and easy-to-understand User Interface and is typically used for signal, text and image processing. Although it has its own set of analysis tools, it can also easily be integrated with custom tools. Some of the already available toolkits include signal-analysis toolkit, an image-manipulation toolkit, etc. Besides, it also checks the data types and reports the usage of any incompatible tools. It also reports errors, if any, as well as useful debug messages in order to resolve them. It also helps track serious bugs, so that the program does not crash. It has two modes of representing the data - a text-editor window or a graph-display window. The graph-display window has the added advantage of being able to zoom in on particular features. Triana is specially useful for automating the repetitive tasks, like finding-and-replacing a character or a string.

70.1.9 Trident

Apache Trident is a “high-level abstraction for doing realtime computing on top of [Apache] Storm” [220]. Similarly to Apache Storm, Apache Trident was developed by Twitter. Furthermore, Trident as a tool “allows you to seamlessly intermix high throughput (millions of messages per second), stateful stream processing with low latency distributed querying” [220]. The five kinds of operations in Trident are described as “Operations that apply locally to each partition and cause no network transfer”, “repartitioning operations that repartition a stream but otherwise don’t change the contents (involves network transfer)”, “aggregation operations that do network transfer as part of the operation”, “operations on grouped streams” and “merges and joins” citewww-trident-overview. These five kinds of operations (i.e. joins, aggregations, grouping, functions, and filters) and the general concepts of Apache Trident are described as similar to “high level batch processing tools like Pig or Cascading” [220].

-- check content --

70.1.10 BioKepler

BioKepler is a Kepler module of scientific workflow components to execute a set of bioinformatics tools using distributed execution patterns [102]. It contains a specialized set of actors called “bioActors” for running bioinformatic tools, directors providing distributed data-parallel (DPP) execution on Big Data platforms such as Hadoop and Spark they are also configurable and reusable [101]. BioKepler contains over 40 example workflows that demonstrate the actors and directors [390].

-- check content --

70.1.11 Galaxy

Ansible Galaxy is a website platform and command line tool that enables users to discover, create, and share community developed roles. Users’ GitHub accounts are used for authentication, allowing users to import roles to share with the ansible community. A description of how Ansible roles are encapsulated and reusable tools for organizing automation content is available in [227]. Thus a role contains all tasks, variables, and handlers that are necessary to complete that role. Roles are depicted as the most powerful part of Ansible as they keep playbooks simple and readable [291]. “They provide reusable definitions that you can include whenever you need and customize with any variables that the role exposes.” GitHub hosts the project documents for Ansible Galaxy [228].

-- check content --

70.1.12 Jupyter and IPython

The Jupyter Notebook is a language-agnostic HTML notebook web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text [513]. The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results [242]. The Jupyter notebook combines two components:

1. A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.
2. Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

Notebooks may be exported to a range of static formats, including HTML (for example, for blog posts), reStructuredText, LaTeX, PDF, and slide shows, via the nbconvert command [243]. Notebook documents contains the inputs and outputs of a interactive session as well as additional text that accompanies the code but is not meant for execution [635]. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects [331]. These documents are internally JSON files and are saved with the .ipynb extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues [691].

70.1.13 Dryad – check content –

Dryad is a general-purpose distributed execution engine for coarse-grain data-parallel applications. Dryad was created with the objective of automatically managing scheduling, distribution, fault tolerance etc. Dryad concentrates on the throughput instead of latency and it assumes that a private data centre is used [176]. It creates a dataflow graph by using computational 'vertices' and communication 'channels'. The computational vertices are written using C++ base classes and objects. During runtime, the dataflow graph is parallelized by distributing the vertices across multiple processor cores on the same computer or different physical computers connected by a network. The Dryad runtime handles this scheduling without any explicit intervention. The data flow from one vertex to another is realized by TCP/IP streams, shared memory, or temporary files. In the directed acyclic graph created by Dryad, each vertex is a program and the edges represent data channels. Each graph is represented as $G = (VG, EG, IG, OG)$, where VG is a sequence of vertices with EG directed edges and two sets IG is a subset of VG and OG is a subset of VG that indicate the input and output vertices respectively [177]. Other technologies used for the same purpose as Dryad include Map Reduce, MPI etc.

-- check content --

70.1.14 Naiad – check content –

Naiad is a distributed system based on computational model called *Timely Dataflow* developed for execution of data-parallel, cyclic dataflow programs [434]. It provides an in-memory distributed dataflow framework which exposes control over data partitioning and enables features like the high throughput of batch processors, the low latency of stream processors, and the ability to perform iterative and incremental computations. The Naiad architecture consists of two main components: (1) incremental processing of incoming updates and (2) low-latency real-time querying of the application state.

Compared to other systems supporting loops or streaming computation, Naiad provides support for the combination of the two, nesting loops inside streaming contexts and indeed other loops, while maintaining a clean separation between the many reasons new records may flow through the computation [642].

This model enriches dataflow computation with timestamps that represent logical points in the computation and provide the basis for an efficient, lightweight coordination mechanism. All the above capabilities in one package allows development of High-level programming models on Naiad which can perform tasks as streaming data analysis, iterative machine learning, and interactive graph mining. On the contrary, it's public reusable low-level programming abstractions leads Naiad to outperforms many other data parallel systems that enforce a single high-level programming model.

-- check content --

70.1.15 Oozie

Oozie is a workflow manager and scheduler. Oozie is designed to scale in a Hadoop cluster. Each job will be launched from a different datanode [336] [694]. Oozie is architected from the ground up for large-scale Hadoop workflow [464]. Scales to meet the demand, provides a multi-tenant service, is secure to protect data and processing, and can be operated cost effective ly. As demand for workflow and the sophistication of applications increase, it must continue to mature in these areas [336].Is well integrated with Hadoop security. Is the only workflow manager with built-in

Hadoop actions, making workflow development, maintenance and troubleshooting easier. It's UI makes it easier to drill down to specific errors in the data nodes. Proven to scale in some of the world's largest clusters [336]. Gets callbacks from MapReduce jobs so it knows when they finish and whether they hang without expensive polling. Oozie Coordinator or allows triggering actions when files arrive at HDFS. Also supported by Hadoop vendors [336].

-- check content --

70.1.16 Tez

Apache Tez is open source distributed execution framework build for writing native YARN application. It provides architecture which allows user to convert complex computation as dataflow graphs and the distributed engine to handle the directed acyclic graph for processing large amount of data. It is highly customizable and pluggable so that it can be used as a platform for various application. It is used by the Apache Hive, Pig as execution engine to increase the performance of map reduce functionality [623]. Tez focuses on running application efficiently on Hadoop cluster leaving the end user to concentrate only on its business logic. Tez provides features like distributed parallel execution on hadoop cluster, horizontal scalability, resource elasticity, shared library reusable components and security features. Tez provides capability to naturally map the algorithm into the hadoop cluster execution engine and it also provides the interface for interaction with different data sources and configurations.

Tez is client side application and just needs Tez client to be pointed to Tez jar libraries path makes it easy and quick to deploy. User can have multiple tez version running concurrently. Tez provides DAG API's which lets user define structure for the computation and Runtime API's which contain the logic or code that needs to be executed in each transformation or task.

70.1.17 Google FlumeJava – check content –

FlumeJava is a java library that allows users to develop and run data parallel pipelines [212]. Its goal is to allow a programmer to express his data-parallel computations in a clear way while simultaneously executing it in the best possible optimized manner. The MapReduce function eases the task of data parallelism. However, a pipeline of MapReduce functions is desired by many real time computation systems. FlumeJava provides these abstractions of data parallel computations by providing support for pipelined execution. To provide optimized parallel execution, FlumeJava defers the execution of these pipelines and instead constructs an execution plan dataflow graph depending on the results needed by each stage of the pipeline. “When the final results of the parallel operations are eventually needed, FlumeJava first optimizes the execution plan, and then executes the optimized operations on appropriate underlying primitives” [120]. FlumeJava library is written on top of the collection framework in Java.

When developing a large pipeline, it is time consuming to find a bug in the later stages and then re-compile and re-evaluate all the operations. FlumeJava library supports a cached execution mode to aid in this scenario. In this mode, it automatically creates temporary files to hold the outputs of each operation it executes [120]. Thus, rather than recomputing all the operations once the pipeline has been rectified to fix all the bugs, it simply reads the output from these temporary files and later deletes them once they are no longer in use.

-- check content --

70.1.18 Crunch

Arvados Crunch is a containerized workflow engine for running complex, multi-part pipelines or workflows in a way that is flexible, scalable, and supports versioning, reproducibility, and provenance while running in virtualized computing environments [157]. The Arvados Crunch framework is designed to support processing very large data batches (gigabytes to terabytes) efficiently [156]. Arvados Crunch increases concurrency by running tasks asynchronously, using many CPUs and network interfaces at once (especially beneficial for CPU-bound and I/O-bound tasks respectively). Crunch also tracks inputs, outputs, and settings so you can verify that the inputs, settings, and sequence of programs you used to arrive at an output is really what you think it was. Crunch ensures that your programs and workflows are repeatable with different versions of your code, OS updates, etc. and allows you to interrupt and resume long-running jobs consisting of many short tasks and maintains timing statistics automatically.

-- check content --

70.1.19 Cascading

Cascading software authored by Chris Wensel is development platform for building the application in Hadoop [116]. It basically act as an abstraction for Apache Hadoop used for creating complex data processing workflow using the scalability of hadoop however hiding the complexity of mapReduce jobs. User can write their program in java without having knowledge of mapReduce. Applications written on cascading are portable.

Cascading Benefits 1. With Cascading application can be scaled as per the data sets. 2. Easily Portable 3. Single jar file for application deployment.

-- check content --

70.1.20 Askalon

Askalan was developed at the University of Innsbruck [506]. It is application development as well as a runtime environment. It allows easy execution of distributed work flow applications in service oriented grids. It uses a Service Oriented Architecture. Also, for its Grid middleware it uses the Globus Toolkit. The work flow applications are developed using Abstract Grid Work flow Language (AGWL). The architecture has various components like the resource broker responsible for brokerage functions like management and reservation, information service for the discovery and organization of resources and data, metascheduler for mapping in the Grid, performance analysis for unification of performance monitoring and integration of the results and the Askalon scheduler.

The Metascheduler is of special significance since it consists of two major components - the workflow converter and the scheduling engine. The former is responsible for conversion of traditional workflows into directed acyclic graphs (DAGs) while the later one is responsible for the scheduling of workflows for various specific tasks. It has a conventional pluggable architecture which allows easy integration of various services. By default, the Heterogeneous Earliest Finish Time (HEFT) is used as the primary scheduling algorithm.

70.1.21 Scalding

70.1.22 e-Science Central – check content –

The e-Science Central is designed to address some of the pitfalls within current Infrastructure as a Service (e.g. Amazon EC2) and Platform as a Service (e.g. force.com) services [294]. For instance, the “majority of potential scientific users, access to raw hardware is of little use as they lack the skills and resources needed to design, develop and maintain the robust, scalable applications they require” and furthermore “current platforms focus on services required for business applications, rather than those needed for scientific data storage and analysis” [294]. It is explained that e-Science Central is a “cloud based platform for data analysis” which is “portable and can be run on Amazon AWS, Windows Azure or your own hardware” [81]. Furthermore e-Science Central is described as a platform, which “provides both Software and Platform as a Service for scientific data management, analysis and collaboration” [294]. This collaborative platform is designed to be scalable while also maintaining ease of use for scientists. Additionally “a project consisting of chemical modeling by cancer researchers” demonstrates how e-Science Central “allows scientists to upload data, edit and run workflows, and share results in the cloud” [294].

-- check content --

70.1.23 Azure Data Factory – check content –

Azure data factory is a cloud based data integration service that can ingest data from various sources, transform/ process data and publish the result data to the data stores. A data management gateway enables access to data on SQL Databases [567]. The data processing is done by It works by creating pipelines to transform the raw data into a format that can be readily used by BI Tools or applications. The services comes with rich visualization aids that aid data analysis. Data Factory supports two types of activities: data movement activities and data transformation activities. Data Movement is a Copy Activity in Data Factory that copies data from a data source to a Data sink [570]. Data Factory supports the following data stores. Data from any source can be written to any sink. Data Transformation: Azure Data Factory supports the following transformation activities such as Map reduce, Hive transformations and Machine learning activities. Data factory is a great tool to analyze web data, sensor data and geo-spatial data.

-- check content --

70.1.24 Google Cloud Dataflow – check content –

Google Cloud Dataflow is a unified programming model and a managed service for developing and executing a wide variety of data processing patterns (pipelines). Dataflow includes SDKs for defining data processing workflows and a Cloud platform managed services to run those workflows on a Google cloud platform resources such as Compute Engine, BigQuery amongst others [252]. Dataflow pipelines can operate in both batch and streaming mode. The platform resources are provided on demand, allowing users to scale to meet their requirements, it’s also optimized to help balance lagging work dynamically.

Being a cloud offering, Dataflow is designed to allow users to focus on devising proper analysis without worrying about the installation and maintaining the underlying data piping and process infrastructure [340].

-- check content --

70.1.25 NiFi (NSA)

NiFi can be described as “An Easy to use, powerful and reliable system to process and distribute data” [70]. This tool aims at automated data flow from sources with different sizes, formats and following different protocols to the centralized location or destination [53].

This comes equipped with an easy use UI where the data flow can be controlled with a drag and a drop. NiFi was initially developed by NSA (called Niagarafiles) using the concepts of flow-based programming and latter submitted to Apache Software foundation [452].

-- check content --

70.1.26 Jitterbit

Jitterbit is an integration tool that delivers a quick, flexible and simpler approach to design, configure, test, and deploy integration solutions [349]. It delivers powerful, flexible, and easy to use integration solutions that connect modern on premise, cloud, social, and mobile infrastructures. Jitterbit employs high performance parallel processing algorithms to handle large data sets commonly found in ETL initiatives [348]. This allows easy synchronization of disparate computing platforms quickly. The Data Cleansing and Smart Reconstruction tools provides complete reliability in data extraction, transformation and loading.

Jitterbit employs a no-code GUI (graphical user interface) and work with diverse applications such as: ETL (extract-transform-load), SaaS (Software as a Service), SOA (service-oriented architecture).

Thus it provides centralized platform with power to control all data. It supports many document types and protocols: XML, web services, database, LDAP, text, FTP, HTTP (S), Flat and Hierarchic file structures and file shares [350]. It is available for Linux and Windows, and is also offered through Amazon EC2 (Amazon Elastic Compute Cloud). Jitterbit Data Loader for Salesforce is a free data migration tool that enables Salesforce administrators automated import and export of data between flat files, databases and Salesforce.

-- check content --

70.1.27 Talend

Talend is Apache Software Foundation sponsor Big data integration tool design to ease the development and integration and management of big data, Talend provides well optimized auto generated code to load transform, enrich and cleanse data inside Hadoop, where one don't need to learn write and maintain Hadoop and spark code. The product has 900+ build-in components feature data integration

Talend features multiple products that simplify the digital transformation tools such as Big data integration, Data integration, Data Quality, Data Preparation, Cloud Integration, Application Integration, Master Data management, Metadata Manager. Talend Integration cloud is secure and managed integration Platform-as-a-service (iPaas), for connecting, cleansing and sharing cloud on premise data.

-- check content --

70.1.28 Pentaho

Pentaho is a business intelligence corporation that provides data mining, reporting, dashboarding and data integration capabilities. Generally, organizations tend to obtain meaningful relationships

and useful information from the data present with them. Pentaho addresses the obstacles that obstruct them from doing so [491]. The platform includes a wide range of tools that analyze, explore, visualize and predict data easily which simplifies blending any data. The sole objective of pentaho is to translate data into value. Being an open and extensible source, pentaho provides big data tools to extract, prepare and blend any data [34]. Along with this, the visualizations and analytics will help in changing the path that the organizations follow to run their business. From spark and hadoop to noSQL, pentaho transforms big data into big insights.

70.1.29 Apatar

-- check content --

70.1.30 Docker Compose

Docker is an open-source container based technology. A container allows a developer to package up an application and all its part including the stack it runs on, dependencies it is associated with and everything the application requires to run within an isolated environment. Docker separates Application from the underlying Operating System in a similar way as Virtual Machines separates the Operating System from the underlying Hardware. Dockerizing an application is very lightweight in comparison with running the application on the Virtual Machine as all the containers share the same underlying kernel, the Host OS should be same as the container OS (eliminating guest OS) and an average machine cannot have more than few VMs running o them.

Docker Machine is a tool that lets you install Docker Engine on virtual hosts, and manage the hosts with docker-machine commands [406]. You can use Machine to create Docker hosts on your local Mac or Windows box, on your company network, in your data center, or on cloud providers like AWS or Digital Ocean. For Docker 1.12 or higher swarm mode is integrated with the Docker Engine, but on the older versions with Machine's swarm option, we can configure a swarm cluster Docker Swarm provides native clustering capabilities to turn a group of Docker engines into a single, virtual Docker Engine. With these pooled resources "you can scale out your application as if it were running on a single, huge computer" as swarm can be scaled upto 1000 Nodes or upto 50,000 containers [171].

-- check content --

70.1.31 KeystoneML

A framework for building and deploying large-scale machine-learning pipelines within Apache Spark. It captures and optimizes the end-to-end large-scale machine learning applications for high-throughput training in a distributed environment with a high-level API [591]. This approach increases ease of use and higher performance over existing systems for large scale learning [591]. It is designed to be a faster and more sophisticated alternative to SparkML, the machine learning framework that's a full member of the Apache Spark club. Whereas SparkML comes with a basic set of operators for processing text and numbers, KeystoneML includes a richer set of operators and algorithms designed specifically for natural language processing, computer vision, and speech processing [592]. It has enriched set of operations for complex domains: vision, NLP, Speech, plus, advanced math And is Integrated with new BDAS technologies: Velox, ml-matrix, soon Planck, TuPAQ and Sample Clean [744].

70.2 Application and Analytics

70.2.1 Mahout

“Apache Mahout software provides three major features: (1) A simple and extensible programming environment and framework for building scalable algorithms (2) A wide variety of premade algorithms for Scala + Apache Spark, H2O, Apache Flink (3) Samsara, a vector math experimentation environment with R-like syntax which works at scale” [403].

-- check content --

70.2.2 MLlib

MLlib is Apache Spark’s scalable machine learning library [420]. Its goal is to make machine learning scalable and easy. MLlib provides various tools such as, algorithms, feature extraction, utilities for data handling and tools for constructing, evaluating, and tuning machine learning pipelines. MLlib uses the linear algebra package Breeze, which depends on netlib-java for optimized numerical processing. MLlib is shipped with Spark and supports several languages which provides functionality for wide range of learning settings. MLlib library includes Java, Scala and Python APIs and is released as a part of Spark project under the Apache 2.0 license [413].

70.2.3 MLbase

MLBase is a distributed machine learning system built with Apache Spark [572], [168]. Machine Learning (ML) and Statistical analysis are tools for extracting insights from big data. MLbase is a tool for execute machine learning algorithms on a scalable platform. It consist of three components MLLib, MLI and ML Optimizer. MLLib was initially developed as a part of MLBase project but is now a part of Apache Spark. MLI is an experimental API for developing ML algorithm and to extract information. It provides high-level abstraction to the core ML algorithms. A prototype is currently implemented against Spark. ML optimizer on the other hand is use to automate the MLI pipeline construction. It solves for the search problem over feature extractors and ML algorithms included in MLI and ML lib. This library is its in early stage and under active development. Several publications are available on distributed machine learning with MLBase [593], [376] and [616].

-- check content --

70.2.4 DataFu

The Apache DataFu project was created out of the need for stable, well-tested libraries for large scale data processing in Hadoop. Apache DataFu consists of two libraries Apache DataFu Pig and Apache DataFu Hourglass [159]. Apache DataFu Pig is a collection of useful user-defined functions for data analysis in Apache Pig. The functions are in areas of Statistics, Bag Operations, Set Operations, Sessions, Sampling, Estimation, Hashing and Link Analysis. Apache DataFu Hourglass is a library for incrementally processing data using Hadoop MapReduce. It is designed to make computations over sliding windows more efficient. For these types of computations, the input data is partitioned in some way, usually according to time, and the range of input data to process is adjusted as new data arrives. Hourglass works with input data that is partitioned by day, as this is a common scheme for partitioning temporal data.

-- check content --

70.2.5 R

R, a GNU project, is a successor to S - a statistical programming language. It offers a range of capabilities - ‘‘programming language, high level graphics, interfaces to other languages and debugging’’. ‘‘R is an integrated suite of software facilities for data manipulation, calculation and graphical display’’. The statistical and graphical techniques provided by R make it popular in the statistical community. The statistical techniques provided include linear and nonlinear modelling, classical statistical tests, time-series analysis, classification and clustering to name a few [523]. The number of packages available in R has made it popular for use in machine learning, visualization, and data operations tasks like data extraction, cleaning, loading, transformation, analysis, modeling and visualization. It’s strength lies in analyzing data using its rich library but falls short when working with very large datasets [510].

70.2.6 pbdR

Programming with Big Data in R (pbdR) is an environment having series of R packages for statistical computing with Big Data using high-performance statistical computation [480]. It uses R, a popular language between statisticians and data miners. *pbdR* focuses on distributed memory system, where data is distributed across several machines and processed in batch mode. It uses MPI for inter process communications. R focuses on single machines for data analysis using an interactive GUI. Currently there are two implementation of pbdR, one Rmpi and another being pbdMpi. Rmpi uses SPMD parallelism while pbdRmpi uses manager/worker parallelism.

-- check content --

70.2.7 Bioconductor – check content –

Bioconductor is an open source and open development platform used for analysis and understanding of high throughput genomic data. Bioconductor is used to analyze DNA microarray, flow, sequencing, SNP, and other biological data. All contributions to Bioconductor are under an open source license. The goals of Bioconductor ‘‘include fostering collaborative development and widespread use of innovative software, reducing barriers to entry into interdisciplinary scientific research, and promoting the achievement of remote reproducibility of research results’’ [238]. The Bioconductor is primarily based on R, as most components of Bioconductor are released in R packages [100]. Extensive documentation is provided for each Bioconductor package as vignettes, which include task-oriented descriptions for the functionalities of each package. Bioconductor has annotation functionality to associate ‘‘genomic data in real time with biological metadata from web databases such as GenBank, Entrez genes and PubMed.’’ Bioconductor also has tools to process genomic annotation data.

-- check content --

70.2.8 ImageJ

ImageJ is a Java-based image processing program developed at the National Institutes of Health (NIH). ImageJ was designed with an open architecture that provides extensibility via Java plugins and recordable macros. Using ImageJ’s built-in editor and a Java compiler, it has enabled to solve many image processing and analysis problems in scientific research from three-dimensional live-cell imaging to radiological image processing. ImageJ’s plugin architecture and built-in development environment has made it a popular platform for teaching image processing [320].

70.2.9 OpenCV

OpenCV stands for Open source Computer Vision. It was designed for computational efficiency and with a strong focus on real-time applications. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. It can take advantage of the hardware acceleration of the underlying heterogeneous compute platform as it is enabled with OpenCL (Open Computing Language) [1]. OpenCV 3.2 is the latest version of the software that is currently available [2].

70.2.10 Scalapack

ScaLAPACK is a library of high-performance linear algebra routines for parallel distributed memory machines. It solves dense and banded linear systems, least squares problems, eigenvalue problems, and singular value problems. It is designed for heterogeneous computing and is portable on any computer that supports Message Passing Interface or Parallel Virtual Machine [559].

ScaLAPACK is an open source software package and is available from netlib via anonymous ftp and the World Wide Web. It contains driver routines for solving standard types of problems, computational routines to perform a distinct computational task, and auxiliary routines to perform a certain subtask or common low-level computation. ScaLAPACK routines are based on block-partitioned algorithms in order to minimize the frequency of data movement between different levels of the memory hierarchy.

70.2.11 PetSc

-- check content --

70.2.12 PLASMA MAGMA

PLASMA is built to address the performance shortcomings of the LAPACK and ScaLAPACK libraries on multicore processors and multi-socket systems of multicore processors and their inability to efficiently utilize accelerators such as Graphics Processing Units (GPUs). Real arithmetic and complex arithmetic are supported in both single precision and double precision. PLASMA has been designed by restructuring the software to achieve much greater efficiency, where possible, on modern computers based on multicore processors. PLASMA does not support band matrices and does not solve eigenvalue and singular value problems. Also, PLASMA does not replace ScaLAPACK as software for distributed memory computers, since it only supports shared-memory machines [112] [501]. Recent activities of major chip manufacturers, such as Intel, AMD, IBM and NVIDIA, make it more evident than ever that future designs of microprocessors and large HPC systems will be hybrid/heterogeneous in nature, relying on the integration (in varying proportions) of two major types of components: 1. Many-cores CPU technology, where the number of cores will continue to escalate because of the desire to pack more and more components on a chip while avoiding the power wall, instruction level parallelism wall, and the memory wall; 2. Special purpose hardware and accelerators, especially Graphics Processing Units (GPUs), which are in commodity production, have outpaced standard CPUs in floating point performance in recent years, and have become as easy, if not easier to program than multicore CPUs [173, 282]. While the relative balance between these component types in future designs is not clear, and will likely to vary over time, there seems to be no doubt that future generations of computer systems, ranging from laptops to supercomputers, will consist of a composition of heterogeneous components [378, 650, 651].

-- check content --

70.2.13 Azure Machine Learning

Azure Machine Learning is a cloud based service that can be used to do predictive analytics, machine learning or data mining. It has features like in-built algorithm library, machine learning studio and a web service [275]. In built algorithm library has implementation of various popular machine learning algorithms like decision tree, SVM, linear regression, neural networks etc. Machine learning studio facilitates creation of predictive models using graphical user interface by dragging, dropping and connecting of different modules that can be used by people with minimal knowledge in the machine learning field. Machine learning studio is a free service for basic version and comes with a monthly charge for advanced versions. Apart from building models, studio also has options to do preprocessing like clean, transform and normalize the data. Webservice provides option to deploy the machine learning algorithm as ready to consume APIs that can be reused in future with minimal effort and can also be published.

70.2.14 Google Prediction API & Translation API

Google Prediction API & Translation API are part of Cloud ML API family with specific roles. Below is a description of each and their use.

Google Prediction API provides pattern-matching and machine learning capabilities. Built on HTTP and JSON, the prediction API uses training data to learn and consecutively use what has been learned to predict a numeric value or choose a category that describes new pieces of data. This makes it easier for any standard HTTP client to send requests to it and parse the responses. The API can be used to predict what users might like, categorize emails as spam or non-spam, assess whether posted comments sentiments are positive or negative or how much a user may spend in a day. Prediction API has a 6 month limited free trial or a paid use for \$10 per project which offers up to 10,000 predictions a day [254].

Google Translation API is a simple programmatic interface for translating an arbitrary string into any supported language. Google Translation API is highly responsive allowing websites and applications to integrate for fast dynamic translation of source text from source language to a target language. Translation API also automatically identifies and translate languages with a high accuracy from over a hundred different languages. Google Translation API is charged at \$20 per million characters making it an affordable localization solution. Translation API is also distributed in two editions, premium edition which is tailored for users with precise long-form translation services like livestream, high volumes of emails or detailed articles and documents. There's also standard edition which is tailored for short, real-time conversations [255].

70.2.15 mlpy

mlpy is an open source python library made for providing machine learning functionality. It is built on top of popular existing python libraries of NumPy, SciPy and GNU scientific libraries (GSL). It also makes extensive use of Cython language. These form the prerequisites for mlpy. [18] explains the significance of its components: NumPy, SciPy provide sophisticated N-dimensional arrays, linear algebra functionality and a variety of learning methods, GSL, which is written in C, provides complex numerical calculation functionality.

mlpy provides a wide range of machine learning methods for both supervised and unsupervised learning problems. mlpy is multiplatform and works both on Python 2 and 3 and is distributed

under GPL3. Mlpy provides both classic and new learning algorithms for classification, regression and dimensionality reduction. A detailed list of functionality is offered by mlpy [421]. Though developed for general machine learning applications, mlpy has special applications in computational biology, particularly in functional genomics modeling.

-- check content --

70.2.16 scikit-learn

Scikit-learn is an open source library that provides simple and efficient tools for data analysis and data mining. It is accessible to everybody and reusable in various contexts. It is built on numpy, Scipy and matplotlib and is commercially usable as it is distributed under many linux distributions [111]. Through a consistent interface, scikit-learn provides a wide range of learning algorithms. Scikits are the names given to the modules for SciPy, a fundamental library for scientific computing and as these modules provide different learning algorithms, the library is named as scikit-learn [564]. It provides an in-depth focus on code quality, performance, collaboration and documentation. Most popular models provided by scikit-learn include clustering, cross-validation, dimensionality reduction, parameter tuning, feature selection and extraction.

70.2.17 PyBrain

The goal of PyBrain is to provide flexible, easy to-use algorithms that are not just simple but are also powerful for machine learning tasks [561]. The algorithms implemented are Long Short-Term Memory (LSTM), policy gradient methods, (multidimensional) recurrent neural networks and deep belief networks. These algorithms include a variety of predefined environments and benchmarks to test and compare algorithms.

PyBrain provides a toolbox for supervised, unsupervised and reinforcement learning as well as black-box and multi-objective optimization as it is much larger than Python libraries.

PyBrain implements many recent learning algorithms and architectures while emphasizing on sequential and nonsequential data and tasks. These algorithms range from areas such as supervised learning and reinforcement learning to direct search / optimization and evolutionary methods. For application-oriented users, PyBrain contains reference implementations of a number of algorithms at the bleeding edge of research and this is in addition to standard algorithms which are not available in Python library. Besides this PyBrain sets itself apart by its versatility for composing custom neural networks architectures that range from (multi-dimensional) recurrent networks to restricted Boltzmann machines or convolutional networks.

-- check content --

70.2.18 ComLearn

Comlearn is a system that makes use of data compression methodologies for mining patterns in a large amount of data. So, it is basically a compression-based machine learning system. For identifying and learning different patterns, it provides a set of utilities which can be used in applying standard compression mechanisms. The most important characteristic of comlearn is its power in mining patterns even in domains that are unrelated. It has the ability to identify and classify the language of different bodies of text [125]. This helps in reducing the work of providing background knowledge regarding a particular classification. It provides such generalization through a library that is written in ANSI C which is portable and works in many environments [125]. Comlearn

provides immediate to access every core functionality in all the major languages as it is designed to be extensible.

70.2.19 DAAL (Intel)

DAAL stands for Data Analytics Acceleration Library. DAAL is software library offered by Intel which is written in C++, python, and Java which implements algorithm for doing efficient and optimized data analysis tasks to solve big-data problems [723]. The library is designed to use data platforms like Hadoop, Spark, R, and Matlab. The important algorithms which DAAL implements are 'Lower Order Moments' which is used to find out max, min standard deviation of a dataset, 'Clustering' which is used to do unsupervised learning by grouping data into unlabelled group. It also include 10-12 other important algorithms.

DAAL supports three processing modes namely batch processing, online processing and distributed processing [326]. Intel DAAL addresses all stages of data analytics pipeline namely pre-processing, transformation, analysis, modelling, validation, and decision making.

-- check content --

70.2.20 Caffe

Caffe is a deep learning framework made with three terms namely expression, speed and modularity [768]. Using Expressive architecture, switching between CPU and GPU by setting a single flag to train on a GPU machine then deploy to commodity cluster or mobile devices. Here the concept of configuration file will come without hard coding the values. Switching between CPU and GPU can be done by setting a flag to train on a GPU machine then deploy to commodity clusters or mobile devices.

It can process over 60 million images per day with a single NVIA k40 GPU. It is being used by academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia.

70.2.21 Torch

Torch is an open source machine learning library, a scientific computing framework [708]. It implements LuaJIT programming language and implements C/CUDA. It implements N-dimensional array. It does routines of indexing, slicing, transposing etc. It has an interface to C language via scripting language LuaJIT. It supports different artificial intelligence models like neural network and energy based models. It is compatible with GPU. The core package of it is *torch*. It provides a flexible N dimensional array which supports basic routings. It has been used to build hardware implementation for data flows like those found in neural networks.

70.2.22 Theano

Theano is a Python library. It was written at the LISA lab. Initially it was created with the purpose to support efficient development of machine learning (ML) algorithms. Theano uses recent GPUs for higher speed. It is used to evaluate mathematical expressions and especially those mathematical expressions that include multi-dimensional arrays. Theano's working is dependent on combining aspects of a computer algebra system and an optimizing compiler. This combination of computer algebra system with optimized compilation is highly beneficial for the tasks which involves complicated mathematical expressions and that need to be evaluated repeatedly as evaluation speed

is highly critical in such cases. It can also be used to generate customized C code for number of mathematical operations. For cases where many different expressions are there and each of them is evaluated just once, Theano can minimize the amount of compilation and analyses overhead [641].

70.2.23 DL4j

DL4j stands for Deeplearning4j [724]. It is a deep learning programming library written for Java and the Java virtual machine (JVM) and a computing framework with wide support for deep learning algorithms. Deeplearning4j includes implementations of the restricted Boltzmann machine, deep belief net, deep autoencoder, stacked denoising autoencoder and recursive neural tensor network, word2vec, doc2vec, and GloVe. These algorithms all include distributed parallel versions that integrate with Apache Hadoop and Spark. It is an open-source software released under Apache License 2.0.

Training with Deeplearning4j occurs in a cluster. Neural nets are trained in parallel via iterative reduce, which works on Hadoop-YARN and on Spark. Deeplearning4j also integrates with CUDA kernels to conduct pure GPU operations, and works with distributed GPUs.

70.2.24 H2O

It is an open source software for big data analysis. It was launched by the Start-up H2O in 2011. It provides an in-memory, distributed, fast and a scalable machine learning and predictive analytics platform that allows the users to build machine learning models on big data [278]. It is written in Java. It is currently implemented in 5000 companies. It provides APIs for R (3.0.0 or later), Python (2.7.x, 3.5.x), Scala (1.4-1.6) and JSON [143]. The software also allows online scoring and modeling on a single platform. It is scalable and has a wide range of OS and language support. It works perfectly on the conventional operating systems, and big data systems such as Hadoop, Cloudera, MapReduce, HortonWorks. It can be used on cloud computing environments such as Amazon and Microsoft Azure [279].

70.2.25 IBM Watson – check content –

IBM Watson is a super computer built on cognitive technology that processes information like the way human brain does by understanding the data in a natural language as well as analyzing structured and unstructured data [317]. It was initially developed as a question and answer tool more specifically to answer questions on the quiz show *Jeopardy* but now it has been seen as helping doctors and nurses in the treatment of cancer. It was developed by IBM's DeepQA research team led by David Ferrucci. With Watson you can create bots that can engage in conversation with you [318]. You can even provide personalized recommendations to Watson by understanding a user's personality, tone and emotion. Watson uses the Apache Hadoop framework in order to process the large volume of data needed to generate an answer by creating in-memory datasets used at run-time. Watson's DeepQA UIMA (Unstructured Information Management Architecture) annotators were deployed as mappers in the Hadoop Map-Reduce framework. Watson is written in multiple programming languages like Java, C++, Prolog and it runs on the SUSE Linux Enterprise Server. Today Watson is available as a set of open source APIs and Software As a Service product as well[318].

-- check content --

70.2.26 Oracle PGX

Numerous information is revealed from graphs. Information like direct and indirect relations or patterns in the elements of the data, can be easily seen through graphs. The analysis of graphs can unveil significant insights. Oracle PGX (Parallel Graph AnalytiX) is a toolkit for graph analysis. “It is a fast, parallel, in-memory graph analytic framework that allows users to load up their graph data, run analytic algorithms on them, and to browse or store the result” [477]. Graphs can be loaded from various sources like SQL and NoSQL databases, Apache Spark and Hadoop [485].

70.2.27 GraphLab

GraphLab is a graph-based, distributed computation, high performance framework for machine learning written in C++ [271]. It is an open source project started by Prof. Carlos Guestrin of Carnegie Mellon University in 2009, designed considering the scale, variety and complexity of real world data. It integrates various high level algorithms such as Stochastic Gradient Descent, Gradient Descent & Locking and provides high performance experience. It includes scalable machine learning toolkits which has implementation for deep learning, factor machines, topic modeling, clustering, nearest neighbors and almost everything required to enhance machine learning models. This framework is targeted for sparse iterative graph algorithms. It helps data scientists and developers easily create and install applications at large scale.

-- check content --

70.2.28 GraphX

GraphX is Apache Spark’s API for graph and graph-parallel computation [62].

GraphX provides:

Flexibility: It seamlessly works with both graphs and collections. GraphX unifies ETL, exploratory analysis, and iterative graph computation within a single system. You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms using the Pregel API.

Speed: Its performance is comparable to the fastest specialized graph processing systems while retaining Apache Spark’s flexibility, fault tolerance, and ease of use.

Algorithms: GraphX comes with a variety of algorithms such as PageRank, Connected Components, Label propagations, SVD++, Strongly connected components and Triangle Count.

It combines the advantages of both data-parallel and graph-parallel systems by efficiently expressing graph computation within the Spark data-parallel framework [766].

It gets developed as a part of Apache Spark project. It thus gets tested and updated with each Spark release.

70.2.29 IBM System G

IBM System G provides a set of Cloud and Graph computing tools and solutions for Big Data [311]. In fact, the G stands for Graph and typically spans a database, visualization, analytics library, middleware and Network Science Analytics tools. It assists the easy creating of graph stores and queries and exploring them via interactive visualizations [312]. Internally, it uses the property graph model for its working. It consists of five individual components - gShell, REST API, Python interface to gShell, Gremlin and a Visualizer. Some of the typical applications wherein it can be

used include Expertise Location, Commerce, Recommendation, Watson, Cybersecurity, etc [394]. However, it is to be noted that the current version does not work in a distributed environment and it is planned that future versions would support it.

70.2.30 GraphBuilder (Intel)

Intel GraphBuilder for Apache Hadoop V2 is a software that is used to build graph data models easily enabling data scientists to concentrate more on the business solution rather than preparing/formatting the data. The software automates (a) Data cleaning, (b) transforming data and (c) creating graph models with high throughput parallel processing using hadoop, with the help of prebuilt libraries. Intel Graph Builder helps to speed up the time to insight for data scientists by automating heavy custom workflows and also by removing the complexities of cluster computing for constructing graphs from Big Data. Intel Graph Building uses Apache Pig scripting language to simplify data preparation pipeline. “Intel Graph Builder also includes a connector that parallelizes the loading of the graph output into the Aurelius Titan open source graph database - which further speeds the graph processing pipeline through the final stage”. Finally being an open source there is a possibility of adding a load of functionalities by various contributors [327].

70.2.31 TinkerPop

ThinkerPop is a graph computing framework from Apache software foundation [68]. Before coming under the Apache project, ThinkerPop was a stack of technologies like Blueprint, Pipes, Frames, Rexters, Furnace and Gremlin where each part was supporting graph-based application development. Now all parts are come under single TinkerPop project repo [179]. It uses Gremlin, a graph traversal machine and language. It allows user to write complex queries (traversal), that can use for real-time transactional (OLTP) queries, graph analytic system (OLAP) or combination of both as in hybrid. Gremlin is written in java [65]. TinkerPop has an ability to create a graph in any size or complexity. Gremlin engine allows user to write graph traversal in Gremlin language, Python, JavaScript, Scala, Go, SQL and SPARQL. It is capable to adhere with small graph which requires a single machine or massive graphs that can only be possible with large cluster of machines, without changing the code.

70.2.32 Parasol

The parasol laboratory is a multidisciplinary research program founded at Texas A&M University with a focus on next generation computing languages. The core focus is centered around algorithm and application development to find solutions to data concentrated problems [486]. The developed applications are being applied in the following areas: computational biology, geophysics, neuroscience, physics, robotics, virtual reality and computer aided drug design (CAD). The program has organized a number of workshops and conferences in the areas such as software, intelligent systems, and parallel architecture.

70.2.33 Dream:Lab

DREAM:Lab stands for “Distributed Research on Emerging Applications and Machines Lab” [483]. DREAM:Lab is centered around distributed systems research to enable expeditious utilization of distributed data and computing systems [483]. DREAM:Lab utilizes the “capabilities of hundreds of personal computers” to allow access to supercomputing resources to average individuals [528]. The DREAM:Lab pursues this goal by utilizing distributed computing [528]. Distributed com-

puting consists of independent computing resources that communicate with each other over a network [164]. A large, complex computing problem is broken down into smaller, more manageable tasks and then these tasks are distributed to the various components of the distributed computing system [164].

-- check content --

70.2.34 Google Fusion Tables

Fusion Tables is a cloud based services, provided by Google for data management and integration. Fusion Tables allow users to upload the data in tabular format using data files like spreadsheet, CSV, KML, .tsv up to 250MB [257]. It used for data management, visualizing data (e.g. pie-charts, bar-charts, lineplot, scatterplot, timelines), sharing of tables, filter and aggregation the data [740]. It allows user to take the data privately, within controlled collaborative group or in public. It allows to integrate the data from different tables from different users or tables. Fusion Table uses two-layer storage, Bigtable and Magastore. The information rows are stored in bigdata table called *Rows*, user can merge the multiple table in to one, from multiple users. ‘‘Megastore is a library on top of bigtable’’ [256]. Data visualization is one the feature, where user can see the visual representation of their data as soon as they upload it. User can store the data along with geospatial information as well.

-- check content --

70.2.35 CINET

A representation of connected entities such as ‘‘physical, biological and social phenomena’’ within a predictive model[127]. Network science has grown its importance understanding these phenomena Cyberinfrastructure is middleware tool helps study Network science, ‘‘by providing unparalleled computational and analytic environment for researcher’’[362].

Network science involves study of graph a large volume which requires high power computing which usually cant be achieve by desktop. Cyberinfrastructure provides cloud based infrastructure (e.g. FutureGrid) as well as use of HPC (e.g. Shadowfax, Pecos). With use of advance intelligent Job mangers, it select the infrastructure smartly suitable for submitted job.

It provides structural and dynamic network analysis, has number of algorithms for ‘‘network analysis such as shortest path, sub path, motif counting, centrality and graph traversal’’. CiNet has number of range of network visualization modules. CiNet is actively being used by several universities, researchers and analyst.

-- check content --

70.2.36 NWB

NWB stands for Network workbench is analysis, modelling and visualization toolkit for the network scientists. It provides an environment which help scientist researchers and practitioner to get online access to the shared resource environment and network datasets for analysis, modelling and visualization of large scale networking application. User can access this network datasets and algorithms previously obtained by doing lot of research and can also add their own datasets helps in speeding up the process and saving the time for redoing the same analysis [455].

NWB provides advanced tools for users to understand and interact with different types of networks. NWB members are largely the computer scientist, biologist, engineers, social and behavioral

scientist. The platform helps the specialist researchers to transfer the knowledge within the broader scientific and research communities.

-- check content --

70.2.37 Elasticsearch

Elasticsearch is a real time distributed, RESTful search and analytics engine which is capable of performing full text search operations for you [185]. It is not just limited to full text search operations but it also allows you to analyze your data, perform CRUD operations on data, do basic text analysis including tokenization and filtering [187]. For example while developing an E-commerce website, Elasticsearch can be used to store the entire product catalog and inventory and can be used to provide search and autocomplete suggestions for the products. Elasticsearch is developed in Java and is an open source search engine which uses standard RESTful APIs and JSON on top of Apache's Lucene - which is a full text search engine library. Clinton Gormley & Zachary Tong describes elastic search as "A distributed real time document store where every field is indexed and searchable" citeelasticsearch-book. They also mention that "Elastic search is capable of scaling to hundreds of servers and petabytes of structured and unstructured data" [188]. mentions that Elastic search can be used on big data by using the Elasticsearch-Hadoop (ES-Hadoop) connector. ES-Hadoop connector lets you index the Hadoop data into the Elastic Stack to take full advantage of the Elasticsearch engine and returns output through Kibana visualizations [186]. A log parsing engine "Logstash" and analytics and visualization platform *Kibana* are also developed alongside Elasticsearch forming a single package.

-- check content --

70.2.38 Kibana

Kibana is an open source data visualization plugin for Elasticsearch [370]. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data [241]. The combination of Elasticsearch, Logstash, and Kibana (also known as ELK stack or Elastic stack) is available as products or service. Logstash provides an input stream to Elastic for storage and search, and Kibana accesses the data for visualizations such as dashboards [299]. Elasticsearch is a search engine based on Lucene [190]. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time [369] [329].

70.2.39 Logstash

Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice [191]. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.

While Logstash originally drove innovation in log collection, its capabilities extend well beyond that use case. Any type of event can be enriched and transformed with a broad array of input, filter, and output plugins, with many native codecs further simplifying the ingestion process. Logstash accelerates your insights by harnessing a greater volume and variety of data.

70.2.40 Graylog

Graylog is an open source log management tool that allows an organization to assemble, organize and analyze large amounts of data from its network activity. It collects and aggregates events from a group of sources and presents data in a streamlines, simplified interface where one can drill down to significant metrics, identify key relationships, generate powerful data visualizations and derive actionable insights [400]. Graylog allows us to centrally collect and manage log messages of an organization's complete infrastructure [229]. A user can perform search on terabytes of log data to discover number of failed logins, find application errors across all servers or monitor the activity of a suspicious user id. Graylog works on top of Elasticsearch and MongoDB to facilitate this high availability searching. Graylog provides visualization through creation of dashboards that allows a user to build pre-defined views on his data to assemble all of his important data only a single click away [272]. Any search result or metric shall be added as a widget on the dashboard to observe trends in one single location. These dashboards can also be shared with other users in the organization. Based on a user's recent search queries, graylog also allows you to distinguish data that are not searched upon very often and thus can be archived on cost effective storage drives. Users can also add certain trigger conditions that shall alert the system about performance issues, failed logins or exceptions in the flow of the application.

70.2.41 Splunk

Splunk is a platform for big data analytics. It is a software product that enables you to search, analyze, and visualize the machine-generated data gathered from the websites, applications, sensors, devices, and so on, that comprise your IT infrastructure or business [596]. After defining the data source, Splunk indexes the data stream and parses it into a series of individual events that you can view and search. It provides distributed search and MapReduce linearly scales search and reporting. It uses a standard API to connect directly to applications and devices. It was developed in response to the demand for comprehensible and actionable data reporting for executives outside a company's IT department [596].

70.2.42 Tableau

Tableau is a family of interactive data visualization products focused on business intelligence [613]. The different products which tableau has built are: Tableau Desktop, for individual use; Tableau Server for collaboration in an organization; Tableau Online, for Business Intelligence in the Cloud; Tableau Reader, for reading files saved in Tableau Desktop; Tableau Public, for journalists or anyone to publish interactive data online. [614]. Tableau uses VizQL as a visual query language for translating drag-and-drop actions into data queries and later expressing the data visually. Tableau also benefits from an Advanced In-Memory Technology for handling large amounts of data. The strengths of Tableau are mainly the ease of use and speed. However, it has a number of limitations, which the most prominent are unfit for broad business and technical user, being closed-source, no predictive analytical capabilities and no support for expanded analytics.

-- check content --

70.2.43 D3.js

D3.js is a JavaScript library responsible for manipulating documents based on data. D3 helps in making data more interactive using HTML, SVG, and CSS. D3's emphasis on web standards makes it framework independent utilizing the full capabilities of modern browsers, combining powerful

visualization components and a data-driven approach to DOM manipulation [158].

It assists in binding random data to a Document Object Model (DOM), followed by applying data-driven transformations to the document. It is very fast, supports large datasets and dynamic behaviours involving interaction and animation.

70.2.44 **three.js**

Three.js is an API library with about 650 contributions till date, where users can create and display an animated 3D computer graphics in a web browser. It is written in javascript and uses WebGL, HTML5 or SVG. Users can animate HTML elements using CSS3 or even import models from 3D modelling apps [643]. In order to display anything using three.js we need three basic features, which are scene, camera and renderer. This will result in rendering the scene with a camera. In addition to these three features, we can add animation, lights (ambience, spot lights, shadows), objects (lines, ribbons, particles), geometry etc [644].

70.2.45 **Potree**

Potree is an opensource tool powered by WebGL based viewer to visualize data from large point clouds [508]. It started at the TU Wien, institute of Computer Graphics and Algorithms and currently begin continued under the Harvest4D project. Potree relies on reorganizing the point cloud data into an multi-resolution octree data structure which is time consuming. Its efficiency can be improved by using techniques such as divide and conquer as discussed in a conference paper Taming the beast: Free and Open Source massive cloud point cloud web visualization [405]. It has also been widely used in works involving spatio-temporal data where the changes in geographical features are across time [283].

-- check content --

70.2.46 **DC.js**

The “DC.js is a javascript charting library with native crossfilter support, allowing exploration on large multi-dimensional datasets [163]. It uses d3 to render charts in CSS-friendly SVG format. Charts rendered using dc.js are data driven and reactive and therefore provide instant feedback to user interaction.” DC.js library can be used to perform data analysis on both mobile devices and different browsers. Under the dc namespace the following chart classes are included: barChart, boxplot, bubbleChart, bubbleOverlay, compositeChart, dataCount, dataGrid, dataTable, geoChoroplethChart, heatMap, legend, lineChart, numberDisplay, pieChart, rowChart, scatterPlot, selectMenu and seriesChart.

-- check content --

70.2.47 **TensorFlow**

TensorFlow is a platform that provides a software library for expressing and executing machine learning algorithms. TensorFlow has a flexible architecture allowing it to be executed with minimal change to many heterogeneous systems such as CPUs and GPUs of mobile devices, desktop machines, and servers [4]. TensorFlow can “express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas”. TensorFlow utilizes data flow graphs in which the “nodes in the graph represent

mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them” [621]. TensorFlow was developed by the Google Brain Team and has a reference implementation that was released on 2015-11-09 under the Apache 2.0 open source license.

-- check content --

70.2.48 CNTK

The Microsoft Cognitive Toolkit - CNTK - is a unified deep-learning toolkit by Microsoft Research. It is in essence an implementation of Computational Network (CN) which supports both CPU and GPU. CNTK supports arbitrary valid computational networks and makes building DNNs, CNNs, RNNs, LSTMS, and other complicated networks as simple as describing the operations of the networks. The toolkit is implemented with efficiency in mind. It removes duplicate computations in both forward and backward passes, uses minimal memory needed and reduces memory reallocation by reusing them. It also speeds up the model training and evaluation by doing batch computation whenever possible [138]. It can be included as a library in your Python or C++ programs, or used as a standalone machine learning tool through its own model description language (BrainScript) [296]. Latest Version:2017-02-10. V 2.0 Beta 11 Release

70.3 Application Hosting Frameworks

70.3.1 Google App Engine

Google App Engine is a cloud computing platform to host your mobile or web applications on Google managed servers. Google App Engine provides automatic scaling for web applications, i.e it automatically allocates more resources to the application upon increase in the number of requests. It gives developers the freedom to focus on developing their code and not worry about the infrastructure. Google App Engine provides built-in services and APIs such as load balancing, automated security scanning, application logging, NoSQL datastores, memcache, and a user authentication API, that are a core part to most applications [71].

An App Engine platform can be run in either the Standard or the Flexible environment. Standard environment lays restrictions on the maximum number of resources an application can use and charges a user based on the instance hours used. The flexible environment as the name suggests provides higher flexibility in terms of resources and is charged based on the CPU and disk utilization. The App Engine requires developers to use only its supported languages and frameworks. Supported languages are Java, Python, Ruby, Scala, PHP, GO, Node.js and other JVM oriented languages. The App Engine datastore uses a SQL like syntax called the GQL (Google Query Language) which works with non-relational databases when compared to SQL [72].

70.3.2 AppScale

AppScale is an application hosting platform. This platform helps to deploy and scale the unmodified Google App Engine application, which run the application on any cloud infrastructure in public, private and on premise cluster [74]. AppScale provide rapid, API development platform that can run on any cloud infrastructure. The platform separates the app logic and its service part to have control over application deployment, data storage, resource use, backup and migration. AppScale is based on Google’s App Engine APIs and has support for Python, Go, PHP and Java applications. It supports single and multimode deployment, which will help with large, dataset or CPU. AppScale

allows to deploy app in thee main mode i.e. dev/test, production and customize deployment [75].

70.3.3 Red Hat OpenShift

OpenShift was launched as a PaaS (Platform as a Service) by Red Hat in the Red Hat Summit, 2011 [542]. It is a cloud application development and hosting platform that envisages shifting of the developer's focus to development by automating the management and scaling of applications [539]. Thus, OpenShift enables us to write our applications in any one web development language (using any framework) and it itself takes up the task of running the application on the web[540]. This has its advantages and disadvantages - advantage being the developer doesn't have to worry about how the stuff works internally (as it is abstracted away) and the disadvantage being that he cannot control how it works, again because it is abstracted.

OpenShift is powered by Origin, which is in turn built using Docker container packaging and Kubernetes container cluster [541]. Due to this, OpenShift offers a lot of options, including online, on-premise and open source project options.

-- check content --

70.3.4 Heroku

Heroku is a platform as a service that is used for building, delivering monitoring and scaling applications[293]. It lets you develop and deploy application quickly without thinking about irrelevant problems such as infrastructure. Heroku also provides a secure and scalable database as a service with number of developers' tools like database followers, forking, data clips and automated health checks. It works by deploying to cedar stack, an online runtime environment that supports apps buit in Java, Node.js, Scala, Clojure, Python and PHP[117]. It uses Git for version controlling. It is also tightly integrated with Salesforce, providing seamless and smooth Heroku and Salesforce data synchronization enabling companies to develop and design creative apps that uses both platforms.

-- check content --

70.3.5 Aerobatic

Aerobatic is a platform that allows hosting static websites[14]. It used to be an ad-on for Bitbucket but now Aerobatic is transitioning to standalone CLI (command Line Tool) and web dashboard. Aerobatic allows automatic builds to different branches. New changes to websites can be deployed using aero deploy command which can be executed from local desktop or any of CD tools and services like Jenkins, Codeship, Travis and so on. It also allows users to configure custom error pages and offers authentication which can also be customized. Aerobatic is backed by AWS cloud. Aerobatic has free plan and pro plan options for customers.

-- check content --

70.3.6 AWS Elastic Beanstalk

AWS Elastic Beanstalk is an orchestration service offered from Amazon Web Services which provides user with a platform for easy and quick deployment of their WebApps and services [28]. Amazon Elastic BeanStack automatically handles the deployment details of capacity provisioning by Amazon Cloud Watch, Elastic Load Balancing, Auto-scaling, and application health monitoring

of the WebApps and service [358]. AWS Management Console allows the users to configure an automatic scaling mechanism of AWS Elastic Beanstalk. Elastic Load Balancing enables a load balancer, which automatically spreads the load across all running instances in an auto-scaling group based on metrics like request count and latency tracked by Amazon CloudWatch. Amazon CloudWatch tracks and stores per-instance metrics, including request count and latency, CPU, and RAM utilization. Elastic Beanstalk supports applications developed in Java, PHP, .NET, Node.js, Python, and Ruby as well as supports different container types for each language such as Apache Tomcat for Java applications, Apache HTTP Server for PHP applications Docker, GO and much more for specific languages where the container defines the infrastructure and software stack to be used for a given environment. “AWS Elastic Beanstalk runs on the Amazon Linux AMI and the Windows Server 2012 R2 AMI. Both AMIs are supported and maintained by Amazon Web Services and are designed to provide a stable, secure, and high-performance execution environment for Amazon EC2 Cloud computing” [28].

70.3.7 Azure

Microsoft Corporation (MSFT) markets its cloud products under the *Azure* brand name. At its most basic, Azure acts as an *infrastructure-as-a-service* (IaaS) provider. IaaS virtualizes hardware components, a key differentiation from other *-as-a-service* products. IaaS “abstract[s] the user from the details of infrastructure like physical computing resources, location, data partitioning, scaling, security, backup, etc” [721].

However, Azure offers a host of closely-related tool and products to enhance and improve the core product, such as raw block storage, load balancers, and IP addresses [416]. For instance, Azure users can access predictive analytics, Bots and Blockchain-as-a-Service as well as more-basic computing, networking, storage, database and management components [415] [416]. The Azure website shows twelve major categories under *Products* and twenty *Solution* categories, e.g., e-commerce or Business SaaS apps.

Azure competes against Amazon’s *Amazon Web Service*, even though IBM (*SoftLayer* and *Bluemix*) and Google (*Google Cloud Platform*) offer IaaS to the market [22, 24, 314, 315]. As of January 2017, Azure’s datacenters span 32 Microsoft-defined *regions*, or 38 *declared regions*, throughout the world [416].

-- check content --

70.3.8 Cloud Foundry

It is an open source software with multi cloud application. It is a platform for running applications and services. It was originally developed by VMware and currently owned by Pivotal. It is written in Ruby and Go. It has a commercial version called Pivotal Cloud Foundry (PFC) [743]. Cloud Foundry is available as a stand alone software package, we can also deploy it to Amazon AWS as well as host it on OpenStack server, HP’s Helion or VMware’s vSphere as given in the blog, it delivers quick application from development to deployment and is highly scalable [130]. It has a DevOps friendly workflow. Cloud Foundry changes the way application and services are deployed and reduces the develop to deployment cycle time.

-- check content --

70.3.9 Pivotal

Pivotal Software, Inc. (Pivotal) is a software and services company. It offers multiple consulting and technology services, which includes Pivotal Web Services, which is an agile application hosting service. It has a single step upload feature *cf push*, another feature called Buildpacks lets us push applications written for any language like Java, Grails, Play, Spring, Node.js, Ruby on Rails, Sinatra or Go. Pivotal Web Services also allows developers to connect to 3rd party databases, email services, monitoring and more from the Marketplace. It also offers performance monitoring, active health monitoring, unified log streaming, web console built for team-based agile development [496].

70.3.10 IBM BlueMix

-- check content --

70.3.11 (Ninefold)

The Australian based cloud computing platform has shut down their services since January 30, 2016 [449].

-- check content --

70.3.12 Jelastic

Jelastic (acronym for Java Elastic) is an unlimited PaaS and Container based IaaS within a single platform that provides high availability of applications, automatic vertical and horizontal scaling via containerization to software development clients, enterprise businesses, DevOps, System Admins, Developers, OEMs and web hosting providers [345]. Jelastic is a Platform-as-Infrastructure provider of Java and PHP hosting. It has international hosting partners and data centers. The company can add memory, CPU and disk space to meet customer needs. The main competitors of Jelastic are Google App Engine, Amazon Elastic Beanstalk, Heroku, and Cloud Foundry. Jelastic is unique in that it does not have limitations or code change requirements, and it offers automated vertical scaling, application lifecycle management, and availability from multiple hosting providers around the world [277].

70.3.13 Stackato

Hewlett Packard Enterprise or HPE Helion Stackato is a platform as a service (PaaS) cloud computing solution. The platform facilitates deployment of the user's application in the cloud and will function on top of an Infrastructure as a service (IaaS) [303]. Multiple cloud development is supported across AWS, vSphere, and Helion Openstack. The platform supports the following programming languages: native .NET support, java, Node.js, python, and ruby. This flexibility is advantageous compared to early PaaS solutions which would force the customer into utilizing a single stack. Additionally, this solution has the capacity to support private, public and hybrid clouds [361]. This capability user has to not have to make choices of flexibility over security of sensitive data when choosing a cloud computing platform.

-- check content --

70.3.14 appfog

AppFog can be described as “a platform as a service (PaaS) provider” [682]. Platform as a service provides a platform for the development of web applications without the necessity of purchasing the software and infrastructure that supports it [364]. PaaS provides an environment for the creation of software [364]. The underlying support infrastructure that AppFog provides includes things such as runtime, middleware, o/s, virtualization, servers, storage, and networking [73]. AppFog is based on VMWare’s CloudFoundry project [682]. It gets things such as MySQL, Mongo, Redis, memCache, etc. running and then manages them [660].

-- check content --

70.3.15 CloudBees

Cloudbees provides Platform as a Service (PaaS) solution, which is a cloud service for Java applications [132]. It is used to build, run and manage the web applications. It was created in 2010 by Jenkins. It has a continuous delivery platform for DevOps, and adds an enterprise-grade functionality with an expert level support. Cloudbees is better than the traditional Java platform as it requires no provision of the nodes, clusters, load balancers and databases. In cloudbees the environment is constantly managed and monitored where a metering and scale updating is done on a real time basis. The platform ships with verified security and enhancements assuring less risk for sharing sensitive information. It implies the task of getting the platform accessed by every user using the feature *Jenkins Sprawl* [133].

70.3.16 Engine Yard

A deployment platform with fully managed services that combines high-end clustering resources to run Ruby and Rails applications in the cloud is offered by Engine Yard. It is designed as a platform-as-a-Service for Web application developers using Ruby on Rails, PHP and Node.js who requires the advantages of cloud computing. Amazon cloud is the platform where the Engine Yard perform its operations and accomplishes application stack for its users. Amazon allows as many as eight regions to Engine Yard to deploy its CPU instances in varying capacities such as normal, high memory and high CPU. According to customer requirements multiple software components are configured and processed when an instance is started in Engine Yard.

Engine Yard builds its version on Gentoo Linux and has a non-proprietary approach to its stack. The stack includes HAProxy load balancer, Nginx and Rack Web servers, Passenger and Unicorn app servers, as well as MySQL and PostgreSQL relational databases in addition to Ruby, PHP, and Node.js. The credibility of Engine Yard rests with orchestration and management as developers have an option of performing functions in Amazon cloud. Standard operations management procedures are performed once the systems are configured and deployed. Key operations tasks such as performing backups, managing snapshots, managing clusters, administering databases and load balancing are taken care of by Engine Yard.

Engine Yard users are empowered as they have more control over virtual machine instances. These instances are dedicated instances and are not shared with other users. As the instances are independent every user can exercise greater control over instances without interferences with other users [608].

-- check content --

70.3.17 (CloudControl)

No Longer active as of Feb. 2016 [709]

70.3.18 dotCloud

dotCloud services were shutdown on February 29,2016 [451].

-- check content --

70.3.19 Dokku

-- check content --

70.3.20 OSGi

-- check content --

70.3.21 HUBzero

HUBzero is a collaborative framework which allows creation of dynamic websites for scientific research as well as educational activities. HUBzero lets scientific researchers work together online to develop simulation and modeling tools. These tools can help you connect with powerful Grid computing resources as well as rendering farms [690]. Thus allowing other researchers to access the resulting tools online using a normal web browser and launch simulation runs on the Grid infrastructure without having to download or compile any code. It is a unique framework with simulation and social networking capabilities [409].

70.3.22 OODT

The Apache Object Oriented Data Technology (OODT) is an open source data management system framework. OODT was originally developed at NASA Jet Propulsion Laboratory to support capturing, processing and sharing of data for NASA's scientific archives. OODT focuses on two canonical use cases: Big Data processing and on Information integration. It facilitates the integration of highly distributed and heterogeneous data intensive systems enabling the integration of different, distributed software systems, metadata and data. OODT is written in the Java, and through its REST API used in other languages including Python [463].

70.3.23 Agave

Agave is an open source, application hosting framework and provides a platform-as-a-service solution for hybrid computing [680]. It provides everything ranging from authentication and authorization to computational, data and collaborative services. Agave manages end to end lifecycle of an application's execution. Agave provides an execution platform, data management platform, or an application platform through which users can execute applications, perform operations on their data or simple build their web and mobile applications [15].

Agave's API's provide a catalog with existing technologies and hence no additional appliances, servers or other software needs to be installed. To deploy an application from the catalog, the user needs to host it on a storage system registered with Agave, and submit to agave, a JSON file that

shall contain the path to the executable file, the input parameters, and specify the desired output location. Agave shall read the JSON file, formalize the parameters, execute the user program and dump the output to the requested destination [680].

70.3.24 Atmosphere

Atmosphere is developed by CyVerse (previously named as iPlant Collaborative). It is a cloud-computing platform. It allows one to launch his own “isolated virtual machine (VM) image [80]. It does not require any machine specification. It can be run on any device (tablet/desktop/laptop) and any machine (Linux/Windows/Mac/Unix). User should have a CyVerse account and be granted permission to access to Atmosphere before he can begin using Atmosphere. No subscription is needed. Atmosphere is designed to execute data-intense bioinformatics tasks that may include (a) Infrastructure as a Service (IaaS) with advanced APIs; (b) Platform as a Service (PaaS), and (c) Software as a Service (SaaS). On Atmosphere one has several images of virtual machine and user can launch any image or instance according to his requirements. The images launched by users can be shared among different members as and when required [398].

70.4 High level Programming

70.4.1 Kite

Kite is a programming language designed to minimize the required experience level of the programmer. It aims to allow quick development and running time and low CPU and memory usage. Kite was designed with lightweight systems in mind. On OS X Leopard, the main Kite library is only 88KB, with each package in the standard library weighing in at 13-30KB. The main design philosophy is minimalism - only include the minimum necessary, while giving developers the power to write anything that they can write in other languages. Kite combines both object oriented and functional paradigms in the language syntax. One special feature is its use of the pipe character (|) to indicate function calls, as opposed to the period (.) or arrow (->) in other languages. Properties are still de-referenced using the period [165]. Kite also offers a digital assistant for programmers. Kite offers a product which sits as a sidebar in code editor and enables programmers to search for opensource codes to implement in their codes. It even provides relevant examples/syntax and also tries to spot errors in the programs [677].

70.4.2 Hive

The reason behind development of Hive is making it easier for end users to use Hadoop. Map reduce programs were required to be developed by users for simple to complex tasks. It lacked expressiveness like query language. So, it was a time consuming and difficult task for end users to use Hadoop. For solving this problem Hive was built in January 2007 and open sourced in August 2008. Hive is an open source data warehousing solution which is built on top of Hadoop. It structures data into understandable and conventional database terms like tables, columns, rows and partitions. It supports HiveQL queries which have structure like SQL queries. HiveQL queries are compiled to map reduce jobs which are then executed by Hadoop. Hive also contains Metastore which includes schemas and statistics which is useful in query compilation, optimization and data exploration [76].

-- check content --

70.4.3 HCatalog

-- check content --

70.4.4 Tajo

Apache Tajo is a big data relational and distributed data warehouse system for Apache's Hadoop framework[63]. It uses the Hadoop Distributed File System (HDFS) as a storage layer and has its own query execution engine instead of the MapReduce framework. Tajo is designed to provide low-latency and scalable ad-hoc queries, online aggregation, and ETL (extraction-transformation-loading process) on large-data sets which are stored on HDFS (Hadoop Distributed File System) and on other data sources [64]. Apart from HDFS, it also supports other storage formats as Amazon S3, Apache HBase, Elasticsearch etc. It provides distributed SQL query processing engine and even has query optimization techniques and provides interactive analysis on large-data sets. Tajo is compatible with ANSI/ISO SQL standard, JDBC standard. Tajo can also store data from various file formats such as CSV, JSON,RCFile, SequenceFile, ORC and Parquet. It provides a SQL shell which allows users to submit the SQL queries. It also offers user defined functions to work with it which can be created in python. A Tajo cluster has one master node and a number of worker nodes [64]. The master node is responsible for performing the query planning and maintaining a coordination among the worker nodes. It does this by dividing a query in small task which are assigned to the workers who have a local query engine for executing the queries assigned to them.

-- check content --

70.4.5 Shark

Data Scientists when working on huge data sets try to extract meaning and interpret the data to enhance insight about the various patterns, opportunities, and possibilities that the dataset has to offer [192]. At a traditional EDW (Enterprise Data Warehouse), a simple data manipulation can be performed using SQL queries but we have to rely on other systems to apply the machine learning algorithms on these data sets. Apache Shark is a distributed query engine developed by the open source community whose goal is to provide a unified system for easy data manipulation using SQL and pushing sophisticated analysis towards the data.

Shark is a data Warehouse system built on top of Apache Spark which does the parallel data execution and is also capable of deep data analysis using the Resilient Distributed Datasets (RDD) memory abstraction which unifies the SQL query processing engine with analytical algorithms [192]. Based on this common abstraction, it allows running two query in the same set of workers and share intermediate data. Since RDDs are designed to scale horizontally, it is easy to add or remove nodes to accommodate more data or faster query processing. Thus, it can be scaled to the large number of nodes in a fault-tolerant manner

''Shark is built on Hive Codebase and it has the ability to execute HIVE QL queries up to 100 times faster than Hive without making any change in the existing queries [192]. ''Shark can run both on the Standalone Mode and Cluster-Mode. Shark can answer the queries 40X faster than Apache Hive and can run machine learning algorithms 25X faster than MapReduce programs in Apache Hadoop on large data sets'' [192]. Thus, this new data analysis system performs query processing and complex analytics (iterative Machine learning) at scale and efficiently recovers from the failures.

70.4.6 Phoenix

In the first quarter of 2013, Salesforce.com released its proprietary SQL-like interface and query engine for HBase, *Phoenix*, to the open source community. The company appears to have been motivated to develop Phoenix as a way to (1) increase accessibility to HBase by using the industry-standard query language (SQL); (2) save users time by abstracting away the complexities of coding native HBase queries; and, (3) implementing query best practices by implementing them automatically via Phoenix [366]. Although Salesforce.com initially *open-sourced* it via Github, by May of 2014 it had become a top-level Apache project [718].

Phoenix, written in Java, "compiles [SQL queries] into a series of HBase scans, and orchestrates the running of those scans to produce regular JDBC result sets [61]. " In addition, the program directs compute intense portions of the calls to the server. For instance, if a user queried for the top ten records across numerous regions from an HBase database consisting of a billion records, the program would first select the top ten records for each region using server-side compute resources. After that, the client would be tasked with selecting the overall top ten" [566].

Despite adding an abstraction layer, Phoenix can actually speed up queries because it optimizes the query during the translation process [366]. For example, "Phoenix beats Hive for a simple query spanning 10M-100M rows" [82].

Finally, another program can enhance HBase's accessibility for those inclined towards graphical interfaces. SQuirell only requires the user to set up the JDBC driver and specify the appropriate connection string [612].

-- check content --

70.4.7 Impala

Cloudera Impala is Cloudera's open source massively parallel processing (MPP) SQL query engine for data stored in a computer cluster running Apache Hadoop [136]. It allows users to execute low latency SQL queries for data stored in HDFS and HBase, without any movement or transformation of data. The Apache Hive provides a powerful query mechanism for hadoop users, but the query response time are not acceptable due to Hive's reliance on MapReduce. Impala technology by Cloudera has its MPP query engine written in C++ replacing the Java engine proves to improve the interactive Hadoop queries and interactive query response time for hadoop users [213]. Impala is faster than Hive also because it executes the SQL queries natively without translating them into Hadoop MapReduce jobs, thus taking less time. Impala uses HiveQL as programming interface and also the Impala's Query Exec Engines are co-located with the HDFS data nodes, so that the data nodes and processing tasks are co-located, following the haddops paradigm [213]. Impala can also use Hbase as a data source. Thus, Impala can be considered as an extension to the Apache Hadoop, providing a better performance alternative to Hive-on-top-of-MapReduce model.

Hive and other frameworks built on MapReduce are best suited for long running batch jobs, such as those involving batch processing of Extract, Transform, and Load (ETL) type jobs [136]. The important applications of Impala are when the data is to be partially analyzed or when the same kind of query is to be processed several times from the dataset. When the data is to be partially analyzed, Impala uses parquet as the file format, which is developed by Twitter and Cloudera and it stores data in vertical manner [8]. When Parquet queries the dataset it only reads the column split part files rather than reading the entire dataset as compared to Hive.

70.4.8 MRQL

MapReduce Query Language (MRQL, pronounced miracle) “is a query processing and optimization system for large-scale, distributed data analysis” [58]. MRQL provides a SQL like language for use on Apache Hadoop, Hama, Spark, and Flink. MRQL allows users to perform complex data analysis using only SQL like queries, which are translated by MRQL to efficient Java code. MRQL can evaluate queries in Map-Reduce (using Hadoop), Bulk Synchronous Parallel (using Hama), Spark, and Flink modes [58].

MRQL was created in 2011 by Leonids Fegaras and is currently in the Apache Incubator [769]. All projects accepted by the Apache Software Foundation (ASF) undergo an incubation period until a review indicates that the project meets the standards of other ASF projects [57].

-- check content --

70.4.9 SAP HANA

SAP HANA is in-memory massively distributed platform that consists of three components: analytics, relational ACID compliant database and application[761]. Predictive analytics and machine learning capabilities are dynamically allocated for searching and processing of spatial, graphical, and text data. SAP HANA accommodates flexible development and deployment of data on premises, cloud and hybrid configurations. In a nutshell, SAP HANA acts as a warehouse that integrates live transactional data from various data sources on a single platform [459]. It provides extensive administrative, security features and data access that ensures high data availability, data protection and data quality.

-- check content --

70.4.10 HadoopDB

HadoopDB is a hybrid of parallel database and MapReduce technologies. It approaches parallel databases in performance and efficiency, yet still yields the scalability, fault tolerance, and flexibility of MapReduce systems. It is a free and open source parallel DBMS. The basic idea behind it is to give Hadoop access to multiple single-node DBMS servers (eg. PostgreSQL or MySQL) deployed across the cluster. It pushes as much as possible data processing into the database engine by issuing SQL queries which results in resembling a shared-nothing cluster of machines [281].

HadoopDB is more scalable than currently available parallel database systems and DBMS/MapReduce hybrid systems. It has been demonstrated on clusters with 100 nodes and should scale as long as Hadoop scales, while achieving superior performance on structured data analysis workloads.

70.4.11 PolyBase

“PolyBase is a technology that accesses and combines both non-relational and relational data, all from within SQL Server. It allows you to run queries on external data in Hadoop or Azure Blob storage acts mediator between SQL and non SQL data store it makes the analysis of the relation data and other data that is non structure to tables (Hadoop)” [505]. Unless there is a way to transfer data between the data stores it is always difficult to do so. PolyBase bridges this gap by operating on data that is external to SQL server. It don't require additional software, querying to external can be done with same syntax as querying a database table. This happens transparently behind the scene, no knowledge of Hadoop or Azure is required.

It can query data store in Hadoop using T-SQL, polybase also makes it easy to access the Azure blob data using T-SQL. There is no need for a separate ETL or import tool while importing data from Hadoop, “Azure blob storage or Azure Data Lake into relational tables. It leverages Microsoft’s Columnstore technology and analysis capabilities while importing” [505]. It also archives data into Hadoop Azure blob and data lake store in cost effective way.

Push computation to Hadoop. The query optimizer makes a cost-based decision to push computation to Hadoop and while doing so will improve query performance. It uses statistics on external tables to make the cost-based decision. Pushing computation creates MapReduce jobs and leverages Hadoop’s distributed computational resources. Scale compute resources. SQL Server PolyBase scale-out groups can be used to improve query performance. This enables parallel data transfer between SQL Server instances and Hadoop nodes, and it adds compute resources for operating on the external data.

-- check content --

70.4.12 Pivotal HD/Hawq

Pivotal HDB is the Apache Hadoop native SQL database powered by Apache HAWQ for data science and machine learning workloads [499]. It can be used to gain deeper and actionable insights into data with out the need from moving data to another platform to perform advanced analytics. Few important problems that Pivot HDB address are as follows Quickly unlock business insights with exceptional performance, Integrate SQL BI tools with confidence and Iterate advanced analytics and machine learning in database support. Pivotal HDB comes with an elastic SQL query engine which combines MPP-based analytical performance, robust ANSI SQL compliance and integrated Apache MADlib for machine learning [500].

-- check content --

70.4.13 Presto

Presto is an open-source distributed SQL query engine that supports interactive analytics on large datasets [512]. It allows interfacing with a variety of data sources such as Hive, Cassandra, RDBMSs and proprietary data source. Presto is used at a number of big-data companies such as Facebook, Airbnb and Dropbox. Presto’s performance compares favorably to similar systems such as Hive and Stinger [124].

-- check content --

70.4.14 Google Dremel

Dremel is a scalable, interactive ad-hoc query system for analysis of read-only nested data. By combining multi-level execution trees and columnar data layout, Google Dremel is capable of running aggregation queries over trillion-row tables in seconds [412]. With Dremel, you can write a declarative SQL-like query against data stored in a read-only columnar format efficiently for analysis or data exploration. It’s also possible to write queries that analyze billions of rows, terabytes of data, and trillions of records in seconds. Dremel can be use for a variety of jobs including analyzing web-crawled documents, detecting e-mail spam, working through application crash reports.

70.4.15 Google BigQuery

Google BigQuery is an enterprise data warehouse used for large scale data analytics [98] [97]. A user can store and query massive datasets by storing the data in BigQuery and querying the database using fast SQL queries using the processing power of Google's infrastructure. In Google BigQuery a user can control access to both the project and the data based on the his business needs which gives the ability to others to view and even query the data [97]. BigQuery can scale the database from GigaBytes to PetaBytes. BigQuery can be accessed using a Web UI or a command-line tool or even by making calls to the BigQuery REST API using a variety of client libraries such as Java, .NET pr python. BigQuery can also be accessed using a variety of third party tool. BigQuery is fully managed to get started on its own, so there is no need to deploy any resources such as disks and virtual machines.

Projects in BigQuery are top-level containers in Google Cloud Platform[98]. They contain the BigQuery Data. Each project is referenced by a name and unique ID. Tables contain the data in BigQuery. Each table has a schema that describes field names, types, and other information. Datasets enable to organise and control access to the tables. Every table must belong to a dataset. A BigQuery data can be shared with others by defining roles and setting permissions for organizations, projects, and datasets, but not on the tables within them. BigQuery stores data in the Capacitor columnar dataformat, and offers the standard database concepts of tables, partitions, columns, and rows [488].

-- check content --

70.4.16 Amazon Redshift

Amazon Redshift is a fully managed, petabyte-scale data werehouse service in the cloud. Redshift service manages all of the work of setting up, operating and scaling a data werehouse. AWS Redshift can perform these tasks including provisioning capacity, monitoring and backing up the cluster, and applying patches as well as upgrades to the Redshift's engine [27]. Redshift is built on the top of technology from the Massive Paraller Processing (MPP) data-werehouse company ParAccel which based on PostgreSQL 8.0.2 to PostgreSQL 9.x with capability to handle analytics workloads on large- scale dataset stored by a column-oriented DBMS principle [713].

70.4.17 Drill

Apache Drill is an open source framework that provides schema free SQL query engine for distributed large-scale datasets [175]. Drill has an extensible architecture at its different layers. It does not require any centralized metadata and does not have any requirement for schema specification. Drill is highly useful for short and interactive ad-hoc queries on very large scale data sets. It is scalable to several thousands of nodes. Drill is also capable to query nested data in various formats like JSON and Parquet. It can query large amount of data at very high speed. It is also capable of performing discovery of dynamic schema. A service called 'Drillbit' is at the core of Apache Drill responsible for accepting requests from the client, processing the required queries, and returning all the results to the client. Drill is primarily focused on non-relational datastores, including Hadoop and NoSQL

-- check content --

70.4.18 Kyoto Cabinet

Kyoto Cabinet is a library of routines for managing a database which is a simple data file containing records [381]. Each record in the database is a pair of a key and a value. Every key and value is serial bytes with variable length. Both binary data and character string can be used as a key and a value. Each key must be unique within a database. There is neither concept of data tables nor data types. Records are organized in hash table or B+ tree. Kyoto Cabinet runs very fast. The elapsed time to store one million records is 0.9 seconds for hash database, and 1.1 seconds for B+ tree database. Moreover, the size of database is very small. The overhead for a record is 16 bytes for hash database, and 4 bytes for B+ tree database. Furthermore, scalability of Kyoto Cabinet is great. The database size can be up to 8EB (9.22e18 bytes).

-- check content --

70.4.19 Pig

-- check content --

70.4.20 Sawzall

Google engineers created the domain-specific programming language (DSL) *Sawzall* as a productivity enhancement tool for Google employees. They targeted the analysis of large data sets with flat, but regular, structures spread across numerous servers. The authors designed it to handle “simple, easily distributed computations: filtering, aggregation, extraction of statistics”, etc. from the aforementioned data sets [494].

In general terms, a Sawzall job works as follows: multiple computers each create a Sawzall instance, perform some operation on a single record out of (potentially) petabytes of data, return the result to an aggregator function on a different computer and then shut down the Sawzall instance.

The engineer’s focus on simplicity and parallelization led to unconventional design choices. For instance, in contrast to most programming languages Sawzall operates on one data record at a time; it does not even preserve state between records [549]. Additionally, the language provides just a single primitive result function, the *emit* statement. The emitter returns a value from the Sawzall program to a designated virtual receptacle, generally some type of aggregator. In another example of pursuing language simplicity and parallelization, the aggregators remain separate from the formal Sawzall language (they are written in C++) because “some of the aggregation algorithms are sophisticated and best implemented in a native language [and] [m]ore important[ly] drawing an explicit line between filtering and aggregation enables a high degree of parallelism, even though it hides the parallelism from the language itself” [494].

Important components of the Sawzall language include: *szl*, the binary containing the code compiler and byte-code interpreter that executes the program; the *libszl* library, which compiles and executes Sawzall programs “[w]hen *szl* is used as part of another program, e.g. in a [map-reduce] program”; the Sawzall language plugin, designated *protoc_gen_szl*, which generates Sawzall code when run in conjunction with Google’s own *protoc* protocol compiler; and libraries for intrinsic functions as well as Sawzall’s associated aggregation functionality [21].

-- check content --

70.4.21 Google Cloud DataFlow

Google Cloud DataFlow is a unified programming model that manages the deployment, maintenance and optimization of data processes such as batch processing, ETL etc [253]. It creates a pipeline of tasks and dynamically allocates resources thereby maintaining high efficiency and low latency. These capabilities make it suitable for solving challenging big data problems [253]. Also, google DataFlow overcomes the performance issues faced by Hadoops Mapreduce while building pipelines[410]. The performance of MapReduce started deteriorating while facing multiple petabytes of data whereas Google Cloud Dataflow is apparently better at handling enormous datasets [253]. Additionally Google Dataflow can be integrated with Cloud Storage, Cloud Pub/Sub, Cloud Datastore, Cloud Bigtable, and BigQuery. The unified programming ability is another noteworthy feature which uses Apache Beam SDKs to support powerful operations like windowing and allows correctness control to be applied to batch and stream data processes.

-- check content --

70.4.22 Summingbird

Summingbird can be described as “a library that lets you write MapReduce programs that look like native Scala or Java collection transformations and execute them on a number of well-known distributed MapReduce platforms, including Storm and Scalding” [105]. Summingbird is open-source and is a domain-specific Scala implemented language [106]. It combines online and batch MapReduce computations into one framework [106]. It utilizes the platforms Hadoop for batch and Storm for online process execution [106]. The open-source Hadoop implementation of MapReduce is a tool which those responsible for data management use to handle problems related to big data [106]. Summingbird uses an algebraic structure called a commutative semigroup to perform aggregations of both batch and online processes [106]. A commutative semigroup is a particular type of semigroup “where the associated binary operation is also commutative” [106]. The types of data that Summingbird takes as inputs are streams and snapshots [106]. The types of data Summingbird jobs generate are called stores and sinks [106]. Stores are “an abstract model of a key-value store” while sinks are unaggregated tuples from a producer [106]. Summingbird aims to simplify the process of both batch and online analytics by exploiting “the formal properties of algebraic structures” to integrate the various modes of distributed processing [106].

-- check content --

70.4.23 Lumberyard

It is powerful and full-featured enough to develop triple-A, current-gen console games and is deeply integrated with AWS and Twitch (an online steaming service) [454]. Lumberyard’s core engine technology is based on Crytek’s CryEngine [231]. The goal is “creating experiences that embrace the notion of a player, broadcaster, and viewer all joining together” [454]. Monetization for Lumberyard will come strictly through the use of Amazon Web Services’ cloud computing. If you use the engine for your game, you’re permitted to roll your own server tech, but if you’re using a third-party provider, it has to be Amazon [673].

70.5 Streams

70.5.1 Storm

Apache Storm is an open source distributed computing framework for analyzing big data in real time [332]. refers storm as the Hadoop of real time data. Storm operates by reading real time input data from one end and passes it through a sequence of processing units delivering output at the other end. The basic element of Storm is called topology. A topology consists of many other elements interconnected in a sequential fashion. Storm allows us to define and submit topologies written in any programming language.

Once under execution, a storm topology runs indefinitely unless killed explicitly. The key elements in a topology are the spout and the bolt. A spout is a source of input which can read data from various datasources and passes it to a bolt. A bolt is the actual processing unit that processes data and produces a new output stream. An output stream from a bolt can be given as an input to another bolt [604].

70.5.2 S4

S4 is a distributed, scalable, fault-tolerant, pluggable platform that allows programmers to easily develop applications for processing continuous unbounded streams of data [54]. It is built on similar concept of key-value pairs like the MapReduce. The core platform is written in Java [552]. S4 provides a runtime distributed platform that handles communication, scheduling and distribution across containers. The containers are called S4 nodes. The data is executed and processed on these S4 nodes. These S4 nodes are then deployed on S4 clusters. The user develops applications and deploys them on S4 clusters for its processing. The applications are built as a graph of Processing Elements (PEs) and Stream that interconnects the PEs. All PEs communicate asynchronously by sending events on streams. Events are dispatched to nodes according to their key in the program [54]. All nodes are symmetric with no centralized service and no single point of failure. Additionally there is no limit on the number of nodes that can be supported. [407]. In S4, both the platform and the applications are built by dependency injection, and configured through independent modules.

-- check content --

70.5.3 Samza

Apache Samza is an open-source near-realtime, asynchronous computational framework for stream processing developed by the Apache Software Foundation in Scala and Java [55]. Apache Samza is a distributed stream processing framework. It uses Apache Kafka for messaging, and Apache Hadoop YARN to provide fault tolerance, processor isolation, security, and resource management. Samza processes streams. A stream is composed of immutable messages of a similar type or category. Messages can be appended to a stream or read from a stream. Samza supports pluggable systems that implement the stream abstraction: in Kafka a stream is a topic, in a database we might read a stream by consuming updates from a table, in Hadoop we might tail a directory of files in HDFS. Samza is a stream processing framework. Samza provides a very simple callback-based *process message* API comparable to MapReduce. Samza manages snapshotting and restoration of a stream processor's state. Samza is built to handle large amounts of state (many gigabytes per partition) [555]. Whenever a machine in the cluster fails, Samza works with YARN to transparently migrate your tasks to another machine. Samza uses Kafka to guarantee that messages are processed in the order they were written to a partition, and that no messages are ever lost. Samza is partitioned and distributed at every level. Kafka provides ordered, partitioned, replayable, fault-tolerant

streams. YARN provides a distributed environment for Samza containers to run in. Samza works with Apache YARN, which supports Hadoop's security model, and resource isolation through Linux CGroups [56] [55].

70.5.4 Granules

Granules is used for execution or processing of data streams in distributed environment. When applications are running concurrently on multiple computational resources, granules manage their parallel execution. The MapReduce implementation in Granules is responsible for providing better performance. It has the capability of expressing computations like graphs. Computations can be scheduled based on periodicity or other activity. Computations can be developed in C, C++, Java, Python, C#, R. It also provides support for extending basic Map reduce framework. Its application domains include hand writing recognition, bio informatics and computer brain interface [573].

70.5.5 Neptune

-- check content --

70.5.6 Google MillWheel

MillWheel is a framework for building low-latency data-processing applications. Users specify a directed computation graph and application code for individual nodes, and the system manages persistent state and the continuous flow of records, all within the envelope of the framework's fault-tolerance guarantees. Other streaming systems do not provide this combination of fault tolerance, versatility, and scalability. MillWheel allows for complex streaming systems to be created without distributed systems expertise. MillWheel's programming model provides a notion of logical time, making it simple to write time-based aggregations. MillWheel was designed from the outset with fault tolerance and scalability in mind. In practice, we find that MillWheel's unique combination of scalability, fault tolerance, and a versatile programming model [17].

70.5.7 Amazon Kinesis

Kinesis is Amazon's real time data processing engine. It is designed to provide scalable, durable and reliable data processing platform with low latency[372]. The data to Kinesis can be ingested from multiple sources in different format. This data is further made available by Kinesis to multiple applications or consumers interested in the data. Kinesis provides robust and fault tolerant system to handle this high volume of data. Data sharding mechanism in Kinesis makes it horizontally scalable. Each of these shards in Kinesis process a group of records which are partitioned by the shard key. Each record processed by Kinesis is identified by sequence number, partition key and data blob. Sequence number to records is assigned by the stream. Partition keys are used by partitioner (a hash function) to map the records to the shards i.e. which records should go to which shard. Producers like web servers, client applications, logs push the data to Kinesis whereas Kinesis applications act as consumers of the data from Kinesis engine. It also provides data retention for certain time for example 24 hours default. This data retention window is a sliding window. Kinesis collects lot of metrics which can be used to understand the amount of data being processed by Kinesis. User can use these metrics to do some analytics and visualize the metrics data. Kinesis is one of the tools part of AWS infrastructure and provides its users a complete software-as-a-service. Kinesis [558] in the area of real-time processing provides following key benefits: ease of use, parallel processing, scalable, cost effective, fault tolerant and highly available.

-- check content --

70.5.8 LinkedIn

LinkedIn is a social networking website for Business and employment [741]. LinkedIn has more than 400 million user profiles (as per 10 March 2016 news), and increasing at a rate of 2 new member every second [166]. LinkedIn provides different products like:

- People You May Know - Skill Endorsements - Jobs You May Be Interested In - News Feed Updates

Such products are based on big data. To achieve such big data tasks, LinkedIn has its ecosystem consist of Oracle, Hadoop, Pig, Hive, Azkaban (Workflow), Avro Data, Zookeeper, Aster Data, Data In- Apache Kafka, Data Out- Apache Kafka and Voldemort [166]. LinkedIn uses Hadoop and Aster Data as an analytics layer [522]. LinkedIn partitioned the user's data into separate DB's stored it in XML format. Voldemort is a key lookup system used to store the analytically-derived data for the products like "People You May Know". Voldemort stores the data in key-value form [522]. LinkedIn has exposed REST API to get the user data [395].

70.5.9 Twitter Heron – check content –

Heron is a real-time analytics platform that was developed at Twitter for distributed streaming processing. Heron was introduced at SIGMOD 2015 to overcome the shortcomings of Twitter Storm as the scale and diversity of Twitter data increased. The primary advantages of Heron were: API compatible with Storm: Back compatibility with Twitter Storm reduced migration time [662]. Task-Isolation: Every task runs in process-level isolation, making it easy to debug/ profile. Use of main stream languages: C++, Java, Python for efficiency, maintainability, and easier community adoption. Support for backpressure: dynamically adjusts the rate of data flow in a topology during run-time, to ensure data accuracy. Batching of tuples: Amortizing the cost of transferring tuples. Efficiency: Reduce resource consumption by 2-5x and Heron latency is 5-15x lower than Storm's latency. The architecture of Heron uses the Storm API to submit topologies to a scheduler [663]. The scheduler runs each topology as a job consisting of several containers. The containers run the topology master, stream manager, metrics manager and Heron instances. These containers are managed by the scheduler depending on resource availability.

-- check content --

70.5.10 Databus

-- check content --

70.5.11 Facebook Puma/Ptail/Scribe/ODS

The real time data Processing at Facebook is carried out using the technologies like Scribe, Ptail, Puma, and ODS. While designing the system, facebook primarily focused on the five key decisions that the system should incorporate which were Ease of Use, Performance, Fault-tolerance, Scalability, and Correctness. "The real time data analytics ecosystem at facebook is designed to handle hundreds of Gigabytes of data per second via hundreds of data pipelines and this system handles over 200,000 events per second with a maximum latency of 30 seconds" [200]. Facebook focused on the Seconds of latency while designing the system and not milliseconds as seconds are fast enough to for all the use case that needs to be supported, and it allowed facebook to use

persistent message bus for data transport and this also made the system more fault tolerant and scalable [200]. The large infrastructure of facebook comprises of hundreds of systems distributed across multiple data centers that needs a continuous monitoring to track their health and performance which is done by Operational Data Store (ODS) [123]. ODS comprises of a time series database (TSDB), which is a query service, and a detection and alerting system. ODS's TSDB is built atop the HBase storage system. Time series data from services running on Facebook hosts is collected by the ODS write service and written to HBase.

When the data is generated by the user from their devices, an AJAX request is fired to facebook, and these requests are then written to a log file using Scribe (distributed data transport system), this messaging system collects, aggregates, and delivers high volume of log data with few seconds of latency and high throughput. Scribe stores the data in the HDFS (Hadoop Distributed File System) in a tailing fashion, where the new events are stored in log files and the files are tailed below the current events. The events are then written into the storage HBase on distributed machines. This makes the data available for both batch and real-time processing. Ptail is an internal tool built to aggregate data from multiple Scribe stores. It then tails the log files and pulls data out for processing. Puma is a stream processing system which is the real-time aggregation/storage of data. Puma provides filtering and processing of Scribe streams (with a few seconds delay), usually Puma batches the storage per 1.5 seconds on average and when the last flush completes, then only a new batch starts to avoid the contention issues, which makes it fairly real time.

70.5.12 Azure Stream Analytics

Azure Stream Analytics is a platform that manages data streaming from devices, web sites, infrastructure systems, social media, internet of things analytics, and other sources using real-time event processing engine [87]. Jobs are authored by “specifying the input source of the streaming data, the output sink for the results of your job, and a data transformation expressed in a SQL-like language.” Some key capabilities and benefits include ease of use, scalability, reliability, repeatability, quick recovery, low cost, reference data use, user defined functions capability, and connectivity. Available documentation to get started with Azure Stream Analytics [172]. Azure Stream Analytics has a development project available on github [86].

-- check content --

70.5.13 Floe

-- check content --

70.5.14 Spark Streaming

Spark Streaming is a library built on top of Spark Core which enables Spark to process real-time streaming data. The streaming jobs can be written similar to batch jobs in Spark, using either Java, Scala or Python. The input to Spark Streaming applications can be fed from multiple data sources such as HDFS, Kafka, Flume, Twitter, ZeroMQ, or custom-defined sources. It also provides a basic abstraction called Discretized Streams or DStreams to represent the continuous data streams. Spark's API for manipulating these data streams is very similar to the Spark Core's Resilient Distributed Dataset (RDD) API which makes it easier for users to move between projects with stored and real-time data as the learning curve is short [590]. Spark Streaming is designed to provide fault-tolerance, throughput, and scalability. Examples of streaming data are messages being published to a queue for real-time flight status update or the log files for a production server [594].

-- check content --

70.5.15 Flink Streaming

-- check content --

70.5.16 DataTurbine

Data Turbine is open source engine that allows to stream data from various sources, process it and sink it to different destinations. The streaming sources can be labs, web cams and Java enabled cell phones [749]. The sinks can be visualizations, interfaces and databases. Data Turbine can be used to stream data formats like numbers, text, sound and video.

The Data Turbine middleware provides the cyber-infrastructure that integrates disparate elements of complex distributed real time applicationcite [222]. Data Turbine acts as a middleware black box using which applications and devices can send and receive data. Data Turbine manages the management operations like memory and file management as well as book-keeping and reconnection logic. Data Turbine also provides Android based controller which allows algorithms to run close to sensors.

-- check content --

70.6 Basic Programming Model and Runtime, SPMD, MapReduce

70.6.1 Hadoop

Apache Hadoop is an open source framework written in Java that utilizes distributed storage and the MapReduce programming model for processing of big data. Hadoop utilizes commodity hardware to build fault tolerant clusters. Hadoop was developed based on papers published by Google on the Google File System (2003) and MapReduce (2004) [717].

Hadoop consists of several modules: the Cluster, Storage, Hadoop Distributed File System (HDFS) Federation, Yarn Infrastructure, MapReduce Framework, and the Hadoop Common Package. The Cluster is comprised of multiple machines, otherwise referred to as nodes. Storage is typically in the HDFS. HDFS federation is the framework responsible for this storage layer. YARN Infrastructure provides computational resources such as CPU and memory. The MapReduce layer is responsible for implementing MapReduce [144]. The Hadoop Common Package which includes operating and file system abstractions and JAR files needed to start Hadoop [717].

70.6.2 Spark

Apache Spark which is an open source cluster computing framework has emerged as the next generation big data processing engine surpassing Hadoop MapReduce. “Spark engine is developed for in-memory processing as well a disk based processing. This system also provides large number of impressive high level tools such as machine learning tool M Lib, structured data processing, Spark SQL, graph processing took Graph X, stream processing engine called Spark Streaming, and Shark for fast interactive question device.” The ability of spark to join datasets across various heterogeneous data sources is one of its prized attributes. Apache Spark is not the most suitable data analysis engine when it comes to processing (1) data streams where latency is the most crucial aspect and (2) when the available memory for processing is restricted. “When available memory is

very limited, Apache Hadoop Map Reduce may help better, considering huge performance gap.” In cases where latency is the most crucial aspect we can get better results using Apache Storm [572].

70.6.3 Twister

Twister is a new software tool released by Indiana University, which is an extension to MapReduce architectures currently used in the academia and industry [339]. It supports faster execution of many data mining applications implemented as MapReduce programs. Applications that currently use Twister include: K-means clustering, Google’s page rank, Breadth first graph search, Matrix multiplication, and Multidimensional scaling. Twister also builds on the SALSA team’s work related to commercial MapReduce runtimes, including Microsoft Dryad software and open source Hadoop software. SALSA project work is funded in part by an award from Microsoft, Inc. The architecture is based on pub/sub messaging that enables it to perform faster data transfers, minimizing the overhead of the runtime. Also, the support for long running processes improves the efficiency of the runtime for many iterative MapReduce computations [661] [338] [771].

-- check content --

70.6.4 MR-MPI

MR-MPI stands for Map Reduce-Message Passing Interface is open source library build on top of standard MPI [432]. It basically implements mapReduce operation providing a interface for user to simplify writing mapReduce program. It is written in C++ and needs to be linked to MPI library in order to make the basic map reduce functionality to be executed in parallel on distributed memory architecture. It provides interface for c, c++ and python. Using C interface the library can also be called from Fortrain.

-- check content --

70.6.5 Stratosphere (Apache Flink)

Apache Flink is an open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications. Apache Flink is used in big data application primarily involving analysis of data stored in Hadoop clusters. It also supports a combination of in-memory and disk-based processing as well as handles both batch and stream processing jobs, with data streaming the default implementation and batch jobs running as special-case versions of streaming application [716].

70.6.6 Reef

REEF (Retainable Evaluator Execution Framework) is a scale-out computing fabric that eases the development of Big Data applications on top of resource managers such as Apache YARN and Mesos [113]. It is a Big Data system that makes it easy to implement scalable, fault-tolerant runtime environments for a range of data processing models on top of resource managers. REEF provides capabilities to run multiple heterogeneous frameworks and workflows of those efficiently. REEF contains two libraries, Wake and Tang where Wake is an event-based-programming framework inspired by Rx and SEDA and Tang is a dependency injection framework inspired by Google Guice, but designed specifically for configuring distributed systems.

-- check content --

70.6.7 Disco

a. Disco from discoproject.org represents an implementation of mapreduce for distributed computing that benefits end users by relieving them of the need to handle “difficult technicalities related to distribution such as communication protocols, load balancing, locking, job scheduling, and fault tolerance” [632]. Its designers wrote the software in Erlang, an inherently fault tolerant language. In addition, Disco’s creators chose Erlang because they believe it best meets the software’s need to handle “tens of thousands of tasks in parallel” [633]. Python was used for Disco’s libraries. Finally, Disco supports pipelines, “a linear sequence of stages, where the outputs of each stage are grouped into the input of the subsequent stage” [128]. Its designers implemented Disco’s libraries in Python. Disco originated within Nokia Corp. to handle large data sets. Since then it has proven itself reliable in production environments outside of Nokia [450].

b. DISCO from the research group Service Engineering (SE), serves as “an abstraction layer for OpenStack’s orchestration component [Heat]” SE based DISCO on its prior orchestration framework, Hurtle. The software sets up a computer cluster and deploys the user’s choice of distributed computing architecture onto the cluster based on setup inputs provided by the user [www-discoabout-discoabstractionlayer]. DISCO offers a command line interface via HTTP to directly access OpenStack [www-discodescribed-discoabstractionlayer].

-- check content --

70.6.8 Hama

Apache Hama is a framework for Big Data analytics which uses the Bulk Synchronous Parallel (BSP) computing model, which was established in 2012 as a Top-Level Project of The Apache Software Foundation. It provides not only pure BSP programming model but also vertex and neuron centric programming models, inspired by Google’s Pregel and DistBelief [35]. It avoids the processing overhead of MapReduce approach such as sorting, shuffling, reducing the vertices etc. Hama provides a message passing interface and each superstep in BSP is faster than a full job execution in MapReduce framework, such as Hadoop [487].

70.6.9 Giraph

Apache Giraph is an iterative graph processing system built for big data [48]. It utilizes Hadoop Mapreduce technology for processing graphs [49]. Giraph was initially developed by Yahoo based on the paper published by Google on Pregel [199]. Facebook with some improvements on Giraph could analyze real world graphs up to a scale of a trillion. Giraph can directly interface with HDFS and Hive (As it’s developed in Java) [560].

-- check content --

70.6.10 Pregel

-- check content --

70.6.11 Pegasus

See Section 70.1.4.

70.6.12 Ligra

Ligra is a Light Weight Graph Processing Framework for the graph manipulation and analysis in shared memory system. It is particularly suited for implementing on parallel graph traversal algorithms where only a subset of the vertices are processed in an iteration. The interface is lightweight as it supplies only a few functions. The Ligra framework has two very simple routines, one for mapping over edges and one for mapping over vertices.

The implementations of several graph algorithms like BFS, breadth-first search, betweenness centrality, graph radii estimation, graph-connectivity, PageRank and Bellman-Ford single-source shortest paths efficient and scalable, and often achieve better running times than ones reported by other graph libraries/systems [574]. Although the shared memory machines cannot be scaled to the same size as distributed memory clusters, but the current commodity single unit servers can easily fit graphs with well over a hundred billion edges in the shared memory systems that are large enough for any of the graphs reported in the paper [357].

70.6.13 GraphChi

GraphChi is a disk-based system for computing efficiently on graphs with large number of edges. It uses a well-known method to break large graphs into small parts, and executes data mining, graph mining, machine learning algorithms. GraphChi can process over one hundred thousand graph updates per second, while simultaneously performing computation [383]. GraphChi is a spin-off of the GraphLab. GraphChi brings web-scale graph computation, such as analysis of social networks, available to anyone with a modern laptop

70.6.14 Galois

Galois system was built by intelligent software systems team at University of Texas, Austin. Galois can be decided as “a system that automatically executes ‘Galoized’ serial C++ or Java code in parallel on shared-memory machinesv[668]. It works by exploiting amorphous data-parallelism, which is present even in irregular codes that are organized around pointer-based data structures such as graphs and trees”. By using Galois provided data structures programmers can write serial programs that gives the performance of parallel execution. Galois employs annotations at loop levels to understand correct context during concurrent execution and executes the code that could be run in parallel. The key idea behind Galois is Tao-analysis, in which parallelism is exploited at compile time rather than at run time by creating operators equivalent of the code by employing data driven local computation algorithm [495]. Galois currently supports C++ and Java.

-- check content --

70.6.15 Medusa-GPU

Graphs are commonly used data structures. However, developers may find it challenging to write correct and efficient programs. Furthermore, graph processing is further complicated by irregularities of graph structures. Medusa enables the developers to write sequential C/C++ code. Medusa provides a set of APIs, which embraces a runtime system to automatically execute those APIs in parallel[773]. A number of optimization techniques are implemented to improve the efficiency of graph processing. Experimental results demonstrate that (1) Medusa greatly simplifies implementation of GPGPU programs for graph processing, with many fewer lines of source code written by developers; (2) The optimization techniques significantly improve the performance of the runtime system, making its performance comparable with or better than manually tuned GPU

graph operations [773]. Medusa has proved to be a powerful framework for networked digital audio and video framework and by exploiting the APIs it takes a modular approach to construct complex graph systems [114].

-- check content --

70.6.16 MapGraph

-- check content --

70.6.17 Totem

Totem is a project to overcome the current challenges in graph algorithms. The project is research the Networked Systems Laboratory (NetSysLab) The issue resides in the scale of real world graphs and the inability to process them on platforms other than a supercomputer. Totem is based on a bulk synchronous parallel (BSP) model that can enable hybrid CPU/GPU systems to process graph based applications in a cost effective manner [445].

-- check content --

70.7 Inter process communication Collectives

70.7.1 point-to-point

70.7.2 (a) publish-subscribe: MPI – check content –

see

- <http://www.slideshare.net/Foxsden/high-performance-processing-of-streaming-data>

70.7.3 (b) publish-subscribe: Big Data

Publish/Subscribe (Pub/Sub) is a communication paradigm in which subscribers register their interest as a pattern of events or topics and then asynchronously receive events matching their interest [568]. On the other hand, publishers generate events that are delivered to subscribers with matching interests. In Pub/sub systems, publishers and subscribers need not know each other. Pub/sub technology is widely used for a loosely coupled interaction between disparate publishing data-sources and numerous subscribing data-sinks. The two most widely used pub/sub schemes are - Topic-Based Publish/Subscribe (TBPS) and Content-Based Publish/Subscribe (CBPS) [196].

Big Data analytics architecture are being built on top of a publish/subscribe service stratum, serving as the communication facility used to exchange data among the involved components [195]. Such a publish/subscribe service stratum brilliantly solves several interoperability issues due to the heterogeneity of the data to be handled in typical Big Data scenarios.

Pub/Sub systems are being widely deployed in Centralized datacenters, P2P environments, RSS feed notifications, financial data dissemination, business process management, Social interaction message notifications- Facebook, Twitter, Spotify, etc.

-- check content --

70.7.4 HPX-5

High Performance ParallelX (HPX-5) is an open source, distributed model that provides opportunity for operations to run unmodified on one-to-many nodes [751]. The dynamic nature of the model accommodates effective “computing resource management and task scheduling”. It is portable and performance-oriented. HPX-5 was developed by IU Center for Research in Extreme Scale Technologies (CREST). Concurrency is provided by lightweight control object (LCO) synchronization and asynchronous remote procedure calls. ParallelX component allows for termination detection and supplies per-process collectives. It “addresses the challenges of starvation, latency, overhead, waiting, energy and reliability”. Finally, it supports OpenCL to use distributed GPU and coprocessors. HPX-5 could be compiled on various OS platforms, however it was only tested on several Linux and Darwin (10.11) platforms. Required configurations and environments could be accessed via [752].

-- check content --

70.7.5 Argo BEAST HPX-5 BEAST PULSAR

Search on the internet was not successful.

70.7.6 Harp

Harp is a simple, easy to maintain, low risk and easy to scale static web server that also serves Jade, Markdown, EJS, Less, Stylus, Sass, and CoffeeScript as HTML, CSS, and JavaScript without any configuration and requires low cognitive overhead [286]. It supports the beloved layout/partial paradigm and it has flexible metadata and global objects for traversing the file system and injecting custom data into templates. It acts like a lightweight web server that was powerful enough for me to abandon web frameworks for dead simple front-end publishing. Harp can also compile your project down to static assets for hosting behind any valid HTTP server.

-- check content --

70.7.7 Netty

Netty “is an asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients” [446]. Netty “is more than a collection of interfaces and classes; it also defines an architectural model and a rich set of design patterns” [408]. It is protocol agnostic, supports both connection oriented protocols using TCP and connection less protocols built using UDP. Netty offers performance superior to standard Java NIO API thanks to optimized resource management, pooling and reuse and low memory copying.

-- check content --

70.7.8 ZeroMQ

ZeroMQ is introduced as a software product that can “connect your code in any language, on any platform” by leveraging “smart patterns like pub-sub, push-pull, and router-dealer” to carry “messages across inproc, IPC, TCP, TIPC, [and] multicast” [149]. It is explained that ZeroMQ’s “asynchronous I/O model” causes this “tiny library” to be “fast enough to be the fabric for clustered products” [148]. It is made clear that ZeroMQ is “backed by a large and open source community” with “full commercial support” [149]. In contrast to Message Passing Interface

(i.e. MPI), which is popular among parallel scientific applications, ZeroMQ is designed as a fault tolerant method to communicate across highly distributed systems.

-- check content --

70.7.9 ActiveMQ

Apache ActiveMQ is a powerful open source messaging and Integration Patterns server [11]. It is a message oriented middleware (MOM) for the Apache Software Foundation that provides high availability, reliability, performance, scalability and security for enterprise messaging [610]. The goal of ActiveMQ is to provide standard-based, message-oriented application integration across as many languages and platforms as possible. ActiveMQ implements the JMS spec and offers dozens of additional features and value on top of this specifications. ActiveMQ is used in many scenarios such as heterogeneous application integration, as a replacement for RPC and to loosen the coupling between applications.

70.7.10 RabbitMQ

RabbitMQ is a message broker which allows services to exchange messages in a fault tolerant manner [524]. It provides variety of features which “enables software applications to connect and scale”. Features are: reliability, flexible routing, clustering, federation, highly available queues, multi-protocol, many clients, management UI, tracing, plugin system, commercial support, large community and user base. RabbitMQ can work in multiple scenarios:

1. Simple messaging: producers write messages to the queue and consumers read messages from the queue. This is synonymous to a simple message queue.
2. Producer-consumer: Producers produce messages and consumers receive messages from the queue. The messages are delivered to multiple consumers in round robin manner.
3. Publish-subscribe: Producers publish messages to exchanges and consumers subscribe to these exchanges. Consumers receive those messages when the messages are available in those exchanges.
4. Routing: In this mode consumers can subscribe to a subset of messages instead of receiving all messages from the queue.
5. Topics: Producers can produce messages to a topic multiple consumers registered to receive messages from those topics get those messages. These topics can be handled by a single exchange or multiple exchanges.
6. RPC: In this mode the client sends messages as well as registers a callback message queue. The consumers consume the message and post the response message to the callback queue.

RabbitMQ is based on AMQP (Advanced Message Queuing Protocol) messaging model[458]. AMQP is described as follows “messages are published to exchanges, which are often compared to post offices or mailboxes. Exchanges then distribute message copies to queues using rules called bindings. Then AMQP brokers either deliver messages to consumers subscribed to queues, or consumers fetch/pull messages from queues on demand”

-- check content --

70.7.11 NaradaBrokering

NaradaBrokering is a content distribution infrastructure for voluminous data streams [141]. The substrate places no limits on the size, rate and scope of the information encapsulated within these streams or on the number of entities within the system. The smallest unit of this substrate called as broker, intelligently process and route messages, while working with multiple underlying communication protocols. The major capabilities of NaradaBrokering consists of providing a message oriented middleware (MoM) which facilitates communications between entities (which includes clients, resources, services and proxies thereto) through the exchange of messages and providing a notification framework by efficiently routing messages from the originators to only the registered consumers of the message in question [223]. Also, it provides salient stream oriented features such as their Secure end-to-end delivery, Robust disseminations, jitter reductions.

NaradaBrokering incorporates support for several communication protocol such as TCP, UDP, Multicast, HTTP, SSL, IPSec and Parallel TCP as well as supports enterprise messaging standards such as the Java Message Service, and a slew of Web Service specifications such as SOAP, WS-Eventing, WS-Reliable Messaging and WS-Reliability [140].

-- check content --

70.7.12 QPid

-- check content --

70.7.13 Kafka

FORMAT WRONG

Apache Kafka is a streaming platform, which works based on publish-subscribe messaging system and supports distributed environment.

Kafka lets you publish and subscribe to the messages. Kafka maintains message feeds based on 'topic'. A topic is a category or feed name to which records are published. Kafka's Connector APIs are used to publish the messages to one or more topics, whereas, Consumer APIs are used to subscribe to the topics.

Kafka lets you process the stream of data at real time. Kafka's stream processor takes continual stream of data from input topics, processes the data in real time and produces streams of data to output topics. Kafka's Streams API are used for data transformation.

Kafka lets you store the stream of data in distributed clusters. Kafka acts as a storage system for incoming data stream. As Kafka is a distributed system, data streams are partitioned and replicated across nodes.

Thus, a combination of messaging, storage and processing data stream makes Kafka a 'streaming platform'. It can be used for building data pipelines where data is transferred between systems or applications. Kafka can also be used by applications that transform real time incoming data [217].

-- check content --

70.7.14 Kestrel

Kestrel is a distributed message queue, with added features and bulletproofing, as well as the scalability offered by actors and the Java virtual machine. It supports multiple protocols: memcache:

the memcache protocol; thrift: Apache Thrift-based RPC; text: a simple text-based protocol. Each queue is strictly ordered following the FIFO (first in, first out) principle. To keep up with performance items are cached in system memory. Kestrel is more durable as queues are stored in memory for speed, but logged into a journal on disk so that servers can be shutdown or moved without losing any data. When kestrel starts up, it scans the journal folder and creates queues based on any journal files it finds there, to restore state to the way it was when it last shutdown (or was killed or died).

Kestrel uses a pull-based data aggregator system that convey data without prior definition on its destination. So the destination can be defined later on either storage system, like HDFS or NoSQL, or processing system, like storm and spark streaming. Each server handles a set of reliable, ordered message queues. When you put a cluster of these servers together, with no cross communication, and pick a server at random whenever you do a set or get, you end up with a reliable, loosely ordered message queue [367].

70.7.15 JMS

JMS (Java Messaging Service) is a java oriented messaging standard that defines a set of interfaces and semantics which allows applications to send, receive, create, and read messages. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous [702]. JMS overcomes the drawbacks of RMI (Remote Method Invocation) where the sender needs to know the method signature of the remote object to invoke it and RPC (Remote Procedure Call), which is tightly coupled i.e it cannot function unless the sender has important information about the receiver.

JMS establishes a standard that provides loosely coupled communication i.e the sender and receiver need not be present at the same time or know anything about each other before initiating the communication. JMS provides two communication domains. A point-to-point messaging domain where there is one producer and one consumer. On generating message, a producer simply pushes the message to a message queue which is known to the consumer. The other communication domain is publish/subscribe model, where one message can have multiple receivers [351].

70.7.16 AMQP

AMQP stands for Advanced Message Queueing Protocol [32]. AMQP is an open internet protocol that allows secure and reliable communication between applications in different organizations and different applications which are on different platforms. AMQP allows businesses to implement middleware applications interoperability by allowing secure message transfer between the applications on a timely manner. AMQP is mainly used by financial and banking businesses. Other sectors that also use AMQP are Defence, Telecommunication, cloud Computing and so on. Apache Qpid, StormMQ, RabbitMQ, MQLight, Microsoft's Windows Azure Service Bus, IIT Software's SwiftMQ and JORAM are some of the products that implement AMQP protocol.

-- check content --

70.7.17 Stomp

-- check content --

70.7.18 MQTT

Message Queuing Telemetry Transport (MQTT) protocol is an Interprocess communication protocol that could serve as better alternative to HTTP in certain cases [429]. It is based on a publish-subscribe messaging pattern. Any sensor or remote machine can publish its data and any registered client can subscribe the data. A broker takes care of the message being published by the remote machine and updates the subscriber in case of new message from the remote machine. The data is sent in binary format which makes it use less bandwidth. It is designed mainly to cater to the needs to devices that has access to minimal network bandwidth and device resources without affecting reliability and quality assurance of delivery. MQTT protocol has been in use since 1999. One of the notable work is project Floodnet, which monitors river and floodplains through a set of sensors [431].

-- check content --

70.7.19 Marionette Collective

It is basically a framework for management of a system where the systems undergo an organized coordination resulting in an automated deployment of systems which creates an orderly workflow or a parallel wise job execution. It doesn't rely on central inventories such as SSH and uses tools such as Middleware [411]. This gives an advantage of delivering a very scalable and quick execution environment. Mcollective gives us a huge advantage of working with a large number of servers, it uses publish/subscribe middleware for communicating with many hosts at once in a parallel manner. Mcollective allows us to interact with a cluster of servers at the same time, it allows us to use a simple command line to call remote agents and there isn't a centralized inventory. Mcollective uses a broadcast paradigm to distribute the requests, where all the servers receives the request at the same time which are also attached with a filter. The servers which match the filter will act on these requests.

-- check content --

70.7.20 Public Cloud: Amazon SNS

Amazon SNS is an Inter process communication service which gives the user simple, end-to-end push messaging service allowing them to send messages, alerts, or notifications. Amazon SNS can be used to send a directed message intended for an entity or to broadcast messages to list of selected entities [585]. It is an easy to use and cost effective mechanism to send push messages. Amazon SNS is compatible to send push notifications to iOS, Windows, Fire OS and Android OS devices.

The SNS system architecture consists of four elements: (1) Topics, (2) Owners, (3) Publishers, and (4) Subscribers [586]. Topics are events or access points that identifies the subject of the event and can be accessed by a unique identifier (URI). Owners create topics and control all access to the topic and define the corresponding permission for each topic. Subscribers are clients (applications, end-users, servers, or other devices) that want to receive messages or notifications on specific topics of interest to them. Publishers send messages to topics. SNS matches the topic with the list of subscribers interested in the topic, and delivers the message to them.

Amazon SNS follows pay as per usage. In general it is \$0.50 per 1 million Amazon SNS Requests [587]. Amazon SNS supports notifications over multiple transport protocols such as HTTP/HTTPS, Email/Email-JSON, SQS (Message queue) and SMS. Amazon SNS can be used with other AWS services such as Amazon SQS, Amazon EC2 and Amazon S3.

-- check content --

70.7.21 Lambda

AWS Lambda is a product from Amazon which facilitates serverless computing [83]. AWS Lambda allows for running the code without the need for provisioning or managing servers, all server management is taken care of by AWS. The code to be run on AWS Lambda is called a server function which can be written in Node.js, Python, Java, C#. Each Lambda function is to be stateless and any persistent data needs are to be handled through storage devices. AWS Lambda function can be set up using the AWS Lambda console where one can set up the function code and specify the event that triggers the functional call. AWS Lambda service supports multiple event sources as identified in [84]. AWS Lambda is designed to use replication and redundancy to provide for high availability both for the service itself and the function it runs. AWS Lambda automatically scales your application by running the code in response to each trigger. The code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload. Billing for AWS Lambda is based on the number of times the code executes and in 100 ms increments of the duration of the processing.

70.7.22 Google Pub Sub

Google Pub/Sub provides an asynchronous messaging facility which assists the communication between independent applications [268]. It works in real time and helps keep the two interacting systems independent. It is the same technology used by many of the Google apps like Gmail, Ads, etc. and so integration with them becomes very easy. Some of the typical features it provides are: (1) Push and Pull - Google Pub/Sub integrates quickly and easily with the systems hosted on the Google Cloud Platform thereby supporting one-to-many, one-to-one and many-to-many communication, using the push and pull requests. (2) Scalability - It provides high scalability and availability even under heavy load without any degradation of latency. This is done by using a global and highly scalable design. (3) Encryption - It provides security by encryption of the stored data as well as that in transit. Other than these important features, it provides some others as well, like the usage of RESTful APIs, end-to-end acknowledgement, replicated storage, etc [267].

70.7.23 Azure Queues

Azure Queues storage is a Microsoft Azure service, providing inter-process communication by message passing [577]. A sender sends the message and a client receives and processes them. The messages are stored in a queue which can contain millions of messages, up to the total capacity limit of a storage account [85]. Each message can be up to 64 KB in size. These messages can then be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS. Similar to the other message queue services, Azure Queues enables decoupling of the components [659]. It runs in an asynchronous environment where messages can be sent among the different components of an application. Thus, it provides an efficient solution for managing workflows and tasks. The messages can remain in the queue up to 7 days, and afterwards, they will be deleted automatically.

70.7.24 Event Hubs

Azure Event Hubs is a hyper-scale telemetry ingestion service. It collects, transforms, and stores millions of events. As a distributed streaming platform, it offers low latency and configurable time retention enabling one to ingest massive amounts of telemetry into the cloud and read the data from multiple applications using publish-subscribe semantics [414]. It is a highly scalable data streaming

platform. Data sent to an Event Hub can be transformed and stored using any real-time analytics provider or batching/storage adapters. With the ability to provide publish-subscribe capabilities, Event Hubs serves as the “on ramp” for Big Data.

-- check content --

70.8 In-memory databases/caches

70.8.1 Gora (general object from NoSQL)

Gora is a in-memory data model which also provides persistence to the big data [269]. Gora provides persistence to different types of data stores. Primary goals of Gora are:

1. data persistence 2. indexing 3. data access 4. analysis 5. map reduce support

Unlike ORM models which mostly work with relational databases for example hibernate gora works for most type of data stores like documents, columnar, key value as well as relational. Gora uses beans to maintain the data in-memory and persist it on disk. Beans are defined using apache avro schema. Gora provides modules for each type of data store it supports. The mapping between bean definition and datastore is done in a mapping file which is specific to a data store. Type Gora workflow will be:

1. define the bean used as model for persistence 2. use gora compiler to compile the bean 3. create a mapping file to map bean definition to datastore 4. update gora.properties to specify the datastore to use 5. get an instance of corresponding data store using datastore factory.

Gora has a query interface to query the underlying data store. Its configuration is stored in gora.properties which should be present in classpath. In the file you can specify default data store used by Gora engine. Gora also has a CI/CD library call GoraCI which is used to write integration tests.

-- check content --

70.8.2 Memcached

Memcached is a free and open-source, high performance, distributed memory object caching system [174]. Although, generic in nature, it is intended for use in speeding up dynamic web applications by reducing the database load.

It can be thought of as a short term memory for your applications. Memcached is an in-memory key-value store for small chunks of arbitrary data from the results of database calls, API calls and page rendering. Its API is available in most of the popular languages. In simple terms, it allows you to take memory from parts of your system where you have more memory than you need and allocate it to parts of your system where you have less memory than you need.

-- check content --

70.8.3 Redis

Redis (Remote Dictionary Server) is an open source, in-memory, key-value database which is commonly referred as a data structure server. “It is called a data structure server and not simply a key-value store because Redis implements data structure which allows keys to contain binary safe strings, hashes, sets, and sortedsets as well as lists” [402]. Redis’s better performance, easy to

use and implement, and atomic manipulation of data structures lends itself to solving problems that are difficult to solve or perform poorly when implemented with traditional relational databases. “Salvator Sanfilippo (Creator of open-source database Redis) makes a strong case that Redis does not need to replace the existing database but is an excellent addition to an enterprise for new functionalities or to solve sometimes intractable problems” [440].

A widely used use pattern for Redis is an in-memory cache for web-applications and the other being the use of pattern for REDIS for metric storage of such quantitative data such as the web page usage and user behavior on gamer leaderboards where using a bit operations on strings, Redis very efficiently stores binary information on a particular characteristics [440]. The other popular Redis use pattern is a communication layer between different systems through a publish/subscribe (pub/sub for short), where one can post the message to one or more channels that can be acted upon by other systems that are subscribed to or listening to that channel for incoming messages. The Companies using REDIS includes Twitter to store the timelines of all the user, Pinterest stores the user follower graph, Github, popular web frameworks like Node.js, Django, Ruby-on-Rails etc.

-- check content --

70.8.4 LMDB (key value)

LMDB (Lighting memory-mapped Database) is a high performance embedded transactional database in form of a key-value store [710]. LMDB is designed around virtual memory facilities found in modern operating systems, multi-version concurrency control (MVCC) and single-level store (SLS) concepts. LMDB stores arbitrary key/data pairs as byte arrays, provides a range-based search capability, supports multiple data items for a single key and has a special mode for appending records at the end of the database (MDB_APPEND) which significantly increases its write performance compared to other similar databases.

LMDB is not a relational database and strictly uses key-value store [736]. Key-value databases allows one write at a time, the difference that LMDB highlights is that write transactions do not block readers nor do readers block writes. Also, it does allow multiple applications on the same system to open and use the store simultaneously which helps in scaling up performance [285].

-- check content --

70.8.5 Hazelcast

Hazelcast is a java based, in memory data grid [728]. It is open source software, released under the Apache 2.0 License [288]. Hazelcast enables predictable scaling for applications by providing in memory access to data. Hazelcast uses a grid to distribute data evenly across a cluster. Clusters allow processing and storage to scale horizontally. Hazelcast can run locally, in the cloud, in virtual machines, or in Docker containers. Hazelcast can be utilized for a wide variety of applications. It has APIs for many programming languages including Python, Java, Scala, C++, .NET and Node.js and supports any binary languages through an Open Binary Client Protocol [728].

70.8.6 Ehcache

EHCACHE is an open-source Java-based cache. It supports distributed caching and could scale to hundred of caches. It comes with REST APIs and could be integrated with popular frameworks like Hibernate [183]. It offers storage tiers such that less frequently data could be moved to slower tiers [184]. It's XA compliant and supports two- phase commit and recovery for transactions. It's

developed and maintained by Terracotta and is available under Apache 2.0 license. It conforms to Java caching standard JSR 107.

70.8.7 Infinispan

Infinispan is a highly available, extremely scalable key/value data store and data grid platform. The design perspective of infinispan is exposing a distributed, highly concurrent data structure to make the most use of modern multi-core as well as multi-processor architectures. It is mostly used as a distributed cache, but also can be used as a object database or NoSQL key/value store [330].

Infinispan is mostly used as a cache store. It is predominantly used for applications that are clustered, and requires a cache coherency for data consistency. Infinispan is written in java and is open source. It is fully transactional. Infinispan is used to add clusterability as well as high availability to frameworks. Infinispan has many use-cases, they are: (1) it can be used as a distributed cache (2) Storage for temporal data, like web sessions, (3) Cross-JVM communication, (4) Shared storage, (5) In-memory data processing and analytics and (6) MapReduce Implementation in the In-Memory Data Grid. It is also used in research and academia as a framework for distribution execution and storage [325].

70.8.8 VoltDB

VoltDB is an in-memory database. It is an ACID-compliant RDBMS which uses a shared nothing architecture to achieve database parallelism. It includes both enterprise and community editions. VoltDB is a scale-out NewSQL relational database that supports SQL access from within pre-compiled Java stored procedures. VoltDB relies on horizontal partitioning down to the individual hardware thread to scale, k-safety (synchronous replication) to provide high availability, and a combination of continuous snapshots and command logging for durability (crash recovery) [678]. The in-memory, scale-out architecture couples the speed of traditional streaming solutions with the consistency of an operational database. This gives a simplified technology stack that delivers low-latency response times (1ms) and hundreds of thousands of transactions per second. VoltDB allows users to ingest data, analyze data, and act on data in milliseconds, allowing users to create per-person, real-time experiences [678].

70.8.9 H-Store

H-Store is an in memory and parallel database management system for on-line transaction processing (OLTP). Specifically, it has been illustrated that H-Store is a highly distributed, row-store-based relational database that runs on a cluster on shared-nothing, main memory executor nodes [304]. H-store trends have been described as “the architectural and application shifts have resulted in modern OLTP databases increasingly falling short of optimal performance. In particular, the availability of multiple-cores, the abundance of main memory, the lack of user stalls, and the dominant use of stored procedures are factors that portend a clean-slate redesign of RDBMSs” [302]. The H-store which is a complete redesign has the potential to outperform legacy OLTP databases by a significant factor. H-Store is the first implementation of a new class of parallel DBMS, called NewSQL, that provides the high-throughput and high-availability of NoSQL systems, but without giving up the transactional guarantees of a traditional DBMS [305]. The H-Store system is able to scale out horizontally across multiple machines to improve throughput, as opposed to moving to a more powerful, more expensive machine for a single-node system.

-- check content --

70.9 Object-relational mapping

70.9.1 Hibernate

Hibernate is an open source project which provides object relational persistence framework for applications in Java. It is an Object relational mapping library (ORM) which provides the framework for mapping object oriented model to relational database. It provides a query language, a caching layer and Java Management Extensions (JMX) support. Databases supported by Hibernate includes DB2, Oracle, MySQL, PostgreSQL. To provide persistence services, Hibernate uses database and configuration data. For using hibernate, firstly a java class is created which represents table in the database. Then columns in database are mapped to the instance variables of created Java class. Hibernate can perform database operations like select, insert, delete and update records in table by automatically creating query. Connection management and transaction management are provided by hibernate. Hibernate saves development and debugging time in comparison to JDBC. But it is slower at runtime as it generates many SQL statements at runtime. It is database independent. For batch processing it is advisable to use JDBC over Hibernate [88].

70.9.2 OpenJPA

Apache OpenJPA is a Java persistence project developed by The Apache Software Foundation that can either be used as Plain old Java Object (POJO) or could be used in any Java EE compliant containers. It provides object relational mapping which effectively simplifies the storing of relational dependencies among objects in databases [373]. Kodo, an implementation of Java Data Objects acted as a precursor to the development of OpenJPA [354]. In 2006, BEA Systems donated the majority of the source code of Kodo to The Apache Software Foundation under the name OpenJPA. Being a POJO, OPenJPA can be used without needing to extend prespecified classes, implementing predefined interfaces and inclusion of annotations. OPenJPA can be used in cases where the focus of the project is majorly on business logic and has n<o dependencies on enterprise frameworks. OPenJPA can be implemented across multiple operating systems, on account of its function of cross platform support. It is written in Java and a most recent stable release came out in April 20, 2016 under the version 2.4.1 with Apache License 2.0.

-- check content --

70.9.3 EclipseLink

EclipseLink is an open source persistence Services project from Eclipse foundation. It is a framework which provide developers to interact with data services including database and web services, Object XML mapping etc [707]. This is the project which was developed out of Oracle's Toplink product. The main difference is EclipseLink does not have some key enterprise feature. EclipseLink support a number of persistence standard model like JPA, JAXB, JCA and Service Data Object. Like Toplink, the ORM (Object relational model) is the technique to convert incompatible type system in Object Oriented programming language. It is a framework for storing java object into relational database.

-- check content --

70.9.4 DataNucleus

DataNucleus (available under Apache 2 open source license) is a data management framework in Java. Formerly known as 'Java Persistent Objects' (JPOX) this was relaunched in 2008 as

'DataNucleus'. DataNucleus Access Platform is a fully compliant implementation of the Java Persistent API (JPA) and Java Data Objects (JDO) specifications [701]. It provides persistence and retrieval of data to a number of datastores using a number of APIs, with a number of query languages. In addition to object-relational mapping (ORM) it can also map and manage data from sources other than RDBMS (PostgreSQL, MySQL, Oracle, SQLServer, DB2, H2 etc.) such as Map-based (Cassandra, HBase), Graph-based (Neo4j), Documents (XLS, OOXML, XML, ODF), Web-based (Amazon S3, Google Storage, JSON), Doc-based (MongoDB) and Others (NeoDatis, LDAP). It supports the JPA (Uses JPQL Query language), JDO (Uses JDOQL Query language) and REST APIs [160]. DataNucleus products are built from a sequence of plugins where each of it is an OSGi bundle and can be used in an OSGi environment. Google App Engine uses DataNucleus as the Java persistence layer [355].

-- check content --

70.9.5 ODBC/JDBC

Open Database Connectivity (ODBC) is an open standard application programming interface (API) for accessing database management systems (DBMS) [465]. ODBC was developed by the SQL Access Group and released in September, 1992. Microsoft Windows was the first to provide an ODBC product. Later the versions for UNIX, OS/2, and Macintosh platforms were developed. ODBC is independent of the programming language, database system and platform.

Java Database Connectivity (JDBC) is a API developed specific to the Java programming language. JDBC was released as part of Java Development Kit (JDK) 1.1 on February 19, 1997 by Sun Microsystems [343]. The 'java.sql' and 'javax.sql' packages contain the JDBC classes. JDBC is more suitable for object oriented databases. JDBC can be used for ODBC compliant databases by using a JDBC-to-ODBC bridge.

70.10 Extraction Tools

70.10.1 UIMA

Unstructured Information Management applications (UIMA) provides a framework for content analytics. It searches unstructured data to retrieve specific targets for the user. For example, when a text document is given as input to the system, it identifies targets such as persons, places, objects and even associations. The UIMA architecture can be thought of as four dimensions: 1. Specifies component interfaces in analytics pipeline. 2. Describes a set of Design patterns. 3. Suggests two data representations: an in-memory representation of annotations for high-performance analytics and an XML representation of annotations for integration with remote web services. 4. Suggests development roles allowing tools to be used by users with diverse skills [705].

UIMA uses different, possibly mixed, approaches which include Natural Language Processing, Machine Learning, IR. UIMA supports multimodal analytics which enables the system to process the resource from various points of view. UIMA is used in several software projects such as the IBM Research's Watson uses UIMA for analyzing unstructured data and Clinical Text Analysis and Knowledge Extraction System (Apache cTAKES) which is a UIMA-based system for information extraction from medical records [581].

-- check content --

70.10.2 Tika

“The Apache Tika toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). All of these file types can be parsed through a single interface, making Tika useful for search engine indexing, content analysis, translation, and much more [646].”

70.11 SQL and SQL Services

70.11.1 Oracle

Oracle database is an object-relational database management system by Oracle. Following are some of the key features of Oracle 1. ANSI SQL Compliance 2. Multi-version read consistency 3. Procedural extensions: PL/SQL and Java [476]. Apart from above they are performance related features, including but not limited to: indexes, in-memory, partitioning, optimization. As of today the latest release of Oracle is Oracle Database 12c Release 1: 12.1 (Patch set as of June 2013) [476].

-- check content --

70.11.2 DB2

DB2 is a Relational DataBase Management System (RDBMS). Though initially introduced in 1983 by IBM to run exclusively on its MVS (Multiple Virtual Storage) mainframe platform, it was later extended to other operating systems like UNIX, Windows and Linux. It is used to store, analyze and retrieve the data and is extended with the support of Object-Oriented features and non-relational structures with XML [162]. DB2 server editions include: Advanced Enterprise Server Edition and Enterprise Server Edition (AESE / ESE) designed for mid-size to large-size business organizations, Workgroup Server Edition (WSE) designed for Workgroup or mid-size business organizations, Express -C provides the capabilities of DB2 at no charge and can run on any physical or virtual systems, Express Edition designed for entry level and mid-size business organizations, Enterprise Developer Edition offers single application developer useful to design, build and prototype the applications for deployment on the IBM server. DB2 has APIs for REXX, PL/I, COBOL, RPG, FORTRAN, C++, C, Delphi, .NET CLI, Java, Python, Perl, PHP, Ruby, and many other programming languages. DB2 also supports integration into the Eclipse and Visual Studio integrated development environments [161].

70.11.3 SQL Server

SQL Server is a relational database management system from Microsoft [599]. As of Jan 2017, SQL Server is available in below editions

1. Standard - consists of core database engine
2. Web - low cost edition for web hosting
3. Business Intelligence - includes standard edition and business intelligence tools like PowerPivot, PowerBI, Master Data Services
4. Enterprise - consists of core database engine and enterprise services like cluster manager
5. SQL Server Azure - core database engine integrated with Microsoft Azure cloud platform and available in platform-as-a-service mode citewww-azuresql.

The technical architecture of SQL Server in OLTP (online transaction processing), hybrid cloud and business intelligence modes is explained in detail [419].

-- check content --

70.11.4 SQLite

SQLite is a serverless SQL database engine whose source code resides in the public domain [597]. SQLite databases, including tables, indices, and views, reside on a single file on the disk [597]. It has a compact library, often taking up less than KiB of space, depending on the particular configuration [597]. Performance is the tradeoff with the smaller size, i.e. performance usually runs faster when given more memory [597]. SQLite transactions comply with the ACID (Atomicity, Consistency, Isolation, Durability) properties [597] [550]. SQLite does not require administration or configuration [658]. There are some limitations associated with SQLite, such as the inability to perform Right Outer Joins, read-only views, and access permissions (other than those that are associated with regular file access permissions) SQLite does not compare directly with client/server databases such as MySQL as they are both trying to solve different problems [598] [658]. While database engines such as MySQL aim to provide a shared database, with different access permissions to different individuals/applications, SQLite has the goal of being a local repository of data for applications [598]. While SQLite is not appropriate for every situation, there certainly exists situations where it can prove to be a prudent choice for data management needs [598].

-- check content --

70.11.5 MySQL

MySQL is a relational database management system [436]. SQL is an acronym for Structured Query Language and is a standardized language used to interact with the databases [436]. Databases provide structure to a collection of data while [436]. A database management system allows for the addition, accessing, and processing of the data stored in a database [436]. Relational databases utilize tables that are broken down into columns, representing the various fields of the table, and rows, which correspond to individual entries in the table [301].

-- check content --

70.11.6 PostgreSQL

PostgreSQL is an open-source relational database management system (DBMS). It runs on all the major operating systems like Linux, Mac OSX, Windows and UNIX. It supports the ACID (Atomicity, Consistency, Isolation and Durability) properties of a conventional DBMS. It supports the standard SQL:2008 data types like INTEGER, NUMERIC, etc. besides providing native interfaces for languages such as C++, C, Java and .Net [638].

With the release of its latest version 9.5, it has included new features like the UPSERT capability, Row Level security and multiple features to support Big Data. These new features rolled out in the latest version make PostgreSQL a very strong contender for modern use. UPSERT feature has predominantly been released for the application developers in order to help them simplify their web application and software development. UPSERT is basically a shorthand of “Insert, on conflict update”. Row Level Security (RLS), as the name suggests, enables the database administrators to control which particular rows could be updated by the users. This helps in ensuring that the users do not inadvertently update rows which they are not meant to. Features such as BRIN indexing, Faster sorts, CUBE, ROLLUP and GROUPING SETS, Foreign Data Wrappers and TABLESAMPLE were added as a part of the new Big Data features. Under BRIN indexing (Block Range Indexing), PostgreSQL supports creating small but powerful indexes for large tables. Using a new algorithm called as “abbreviated keys”, PostgreSQL can sort NUMERIC data very quickly. The CUBE, ROLLUP and GROUPING clauses enable the users to use just a single query to create

myriad reports at different levels of summarization. Using the concept of Foreign Data Wrappers (FDWs), PostgreSQL can be used for querying Big Data systems like Cassandra and Hadoop. The TABLESAMPLE clause allows quick statistical sample generation of huge tables without any need to sort them [639].

70.11.7 CUBRID

CUBRID name is deduced from the combination of word CUBE (security within box) and BRIDGE (data bridge). It is an open source Relational DataBase Management System designed in C programming language with high performance, scalability and availability features. During its development by NCL, Korean IT service provider the goal was to optimize database performance for web-applications [154]. Importantly most of the SQL syntax from MySQL and ORACLE can work on cubrid. CUBRID also provides manager tool for database administration and migration tool for migrating the data from DBMS to CUBRID bridging the dbs. CUBRID enterprise version and all the tools are free and suitable database candidate for web-application development.

-- check content --

70.11.8 Galera Cluster

Galera cluster is a type of database clustering which has all multiple masters and works on synchronous replication [230]. At a deeper level, it was created by extending MySQL replication API to provide all support for true multi master synchronous replication. This extended api is called as Write-Set Replication API and is the core of the clustering logic. Each transaction of wsrep API not only contains the record but also other meta-info to requires to commit each node separately or asynchronously. So though it seems synchronous logically but works independently on each node. The approach is also called virtually synchronous replication. This helps in directly read-write on a specific node and can lose a node without handling any complex failover scenarios (zero downtime).

-- check content --

70.11.9 SciDB

SciDB is an open source DBMS based on multi-dimensional array data model and runs on Linux platform [603]. The data store is optimized for mathematical operations such as linear algebra and statistical analysis. The data can be distributed across multiple nodes in a cluster.

The dimensions of the data can be either standard integers or user-defined types. Ragged arrays are also supported. The data is accessed through AQL, a SQL like language designed specifically for array operations. It supports operations such as to filter and join arrays and aggregation over the cell values. It has few similarities to Postgres in terms of user-defined scalar functions and storage manager. Old values of data are updated instead of being deleted to retain different versions of a cell. The arrays are divided into chunks and partitioned across the nodes in the cluster, with provision of caching some of them in the main memory.

-- check content --

70.11.10 Rasdaman

Rasdaman is an specialized database management system which adds capabilities for storage and retrieval of massive multi-dimensional array, such as sensors,image, and statistics data [711]. It is written in C++ language. For example, it can serve 1-D measurement data, 2-D satellite data, 3-D x/y/t image series and x/y/z exploration data, 4-D ocean and climate data, and much more.

Rasdaman servers provides functionality from geo service up to complex analytics which are related to spatio-temporal raster data [529]. It also integrates smoothly with R, OpenLayers, NASA WorldWind etc. via APIs calls. It is massively used in the domains like earth, space, and social science related fields.

-- check content --

70.11.11 Apache Derby

Apache Derby is java based relational database system [628]. Apache Derby has JDBC driver which can be used by Java based applications. Apache derby is part of the Apache DB subproject and licensed under Apache version 2.0.

Derby Embedded Database Engine is the database engine with JDBC and SQL as programming APIs. Client/Server functionality is achieved by Derby network server, it allows connection through TCP/IP using DRDA protocol [629]. ij, database utility makes it possible for SQL scripts to be run on JDBC database. The dblook utility is the schema extraction tool. The sysinfo utility is used for displaying version of Java environment and Derby.

There are two deployment options for Apache Derby, embedded and Derby network server option. In embedded framework, Derby is started and stopped by the single user java application without any administration required. In the case of Derby network server configuration, Derby is started by multi user java application over TCP/IP. Since Apache Derby is written in Java, it runs on any certified JVM (Java Virtual Machine) [627].

-- check content --

70.11.12 Pivotal Greenplum

Pivotal Greenplum is a commercial fully featured data warehouse. It is powered by Greenplum Database an open source initiative. “It is powered by advanced cost-based query optimizer thereby delivering high analytical query performance on large data volumes”. Pivotal Greenplum is uniquely focused on big data analytics [497].

The system consists of a master node, standby master node and segment nodes. The master node consists of the catalog information whereas the data resides on the segment nodes. The segment nodes runs on one or more segments which are modified PostgreSQL databases and are assigned a content identifier. The data is distributed among these segment nodes. The segment node also supports bult loading and unloading. The master node parses, optimizes an SQL query and dispatch it to all segment nodes. Therefore, it provides powerful and rapid analytics on petabyte scale data volumes [498].

70.11.13 Google Cloud SQL

Google Cloud SQL is a fully managed data base as service developed by Google where google manages the backup,patching and replication of the databases etc [261]. Cloud SQL database aims

at developers to focus on app development leaving database administration to a minimum. This can be understood as 'My SQL on Cloud' as most of the features from MySQL 5.7 are directly supported in Cloud SQL. The service is offered with 'Pay per use' providing the flexibility and 'better performance per dollar'. Cloud SQL is scalable up to 16 processor cores and more than 100GB of RAM [262].

-- check content --

70.11.14 Azure SQL

-- check content --

70.11.15 Amazon RDS

According to Amazon Web Services, Amazon Relation Database Service (Amazon RDS) is a web service which makes it easy to setup, operate and scale relational databases in the cloud. Amazon RDS allows to create and use MySQL, Oracle, SQL Server, and PostgreSQL databases in the cloud [538]. Thus, codes, applications and tools used with existing databases can be used with Amazon RDS. The basic components of Amazon RDS include: DB Instances: DB instance is an isolated database environment in the cloud. Regions and availability zones: Region is a data center location which contains Availability Zones [537]. Availability Zone is isolated from failures in other Availability Zones. Security groups: controls access to DB instance by allowing access to IP address ranges or Amazon EC2 instances that is specified. DB parameter groups: manage configuration of DB engine by specifying engine configuration values that are applied to one or more DB instances of the same instance type. DB option groups: Simplifies data management through Oracle Application Express (APEX), SQL Server Transparent Data Encryption, and MySQL memcached support.

-- check content --

70.11.16 Google F1

F1 is a distributed relational database system built at Google to support the AdWords business. It is a hybrid database that combines high availability, the scalability of NoSQL systems like Bigtable, and the consistency and usability of traditional SQL databases. F1 is built on Spanner, which provides synchronous cross-datacenter replication and strong consistency [576].

F1 features include a strictly enforced schema, a powerful parallel SQL query engine, general transactions, change tracking and notification, and indexing, and is built on top of a highly-distributed storage system that scales on standard hardware in Google data centers. The store is dynamically sharded and is able to handle data center outages without data loss [575]. The synchronous cross-datacenter replication and strong consistency results in higher commit latency which can be overcome using hierarchical schema model with structured data types and through smart application design.

70.11.17 IBM dashDB

IBM dashDB is a data warehousing service hosted in cloud, This aims at integrating the data from various sources into a cloud data base. Since the data base is hosted in cloud it would have the benefits of a cloud like scalability and less maintainance. This data base can be configured as 'transaction based' or 'Analytics based' depending on the work load [3]. This is available through

ibm blue mix cloud platform.

dash DB has build in analytics based on IBM Netezza Analytics in the PureData System for Analytics. Because of the build in analytics and support of in memory optimization promises better performance efficiency. This can be run alone as a standalone or can be connected to various BI or analytic tools [316].

-- check content --

70.11.18 N1QL

-- check content --

70.11.19 BlinkDB

-- check content --

70.11.20 Spark SQL

Spark SQL is Apache Spark's module for working with structured data. Spark SQL is a new module that integrates relational processing with Spark's functional programming API [630]. It is used to seamlessly mix SQL queries with Spark programs. Spark SQL lets you query structured data inside Spark programs, using either SQL or a familiar DataFrame API. It offers much tighter integration between relational and procedural processing, through a declarative DataFrame API that integrates with procedural Spark code. Spark SQL reuses the Hive frontend and metastore, giving you full compatibility with existing Hive data, queries, and UDFs by installing it alongside Hive. Spark SQL includes a cost-based optimizer, columnar storage and code generation to make queries fast [9]. At the same time, it scales to thousands of nodes and multi hour queries using the Spark engine, which provides full mid-query fault tolerance.

70.12 NoSQL

70.12.1 Lucene

Apache Lucene is a high-performance, full-featured text search engine library [52]. It is originally written in pure Java but also has been ported to few other languages chiefly python. It is suitable for applications that requires full-text search. One of the key implementation of Lucene is Internet search engines and local, single-site searching. Another important implementation usage is its recommendation system. The core idea of Lucene is to extract text from any document that contains text (not image) field, making it format independent.

-- check content --

70.12.2 Solr

-- check content --

70.12.3 Solandra

Solandra is a highly scalable real-time search engine built on Apache Solr and Apache Cassandra. Solandra simplifies maintaining a large scale search engine, something that more and more appli-

cations need. At its core, Solandra is a tight integration of Solr and Cassandra, meaning within a single JVM both Solr and Cassandra are running, and documents are stored and distributed using Cassandra's data model [341].

Solandra supports most out-of-the-box Solr functionality (search, faceting, highlights), multi-master (read/write to any node). It features replication, sharing, caching, and compaction managed by Cassandra [342].

-- check content --

70.12.4 Voldemort

Project Voldemort, developed by LinkedIn, is a non-relational database of key-value type that supports eventual consistency [676]. The distributed nature of the system allows pluggable data placement and provides horizontal scalability and high consistency. Replication and partitioning of data is automatic and performed on multiple servers. Independent nodes that comprise the server support transparent handling of server failure and ensure absence of a central point of failure. Essentially, Voldemort is a hashtable. It uses APIs for data replication. In memory caching allows for faster operations. It allows cluster expansion with no data rebalancing. When Voldemort performance was benchmarked with the other key-value databases such as Cassandra, Redis and HBase as well as MySQL relational database, the Voldemart's throughput was twice lower than MySQL and Cassandra and six times higher than HBase. Voldemort was slightly underperforming in comparison with Redis [525]. At the same time, it demonstrated consistent linear performance in maximum throughput that supports high scalability. The read latency for Voldemort was fairly consistent and only slightly underperformed Redis. Similar tendency was observed with the read latency that puts Voldemort in the cluster of databases that require good read-write speed for workload operations. However, the same authors noted that Voldemort required creation of the node specific configuration and optimization in order to successfully run a high throughput tests. The default options were not sufficient and were quickly saturated that stall the database.

-- check content --

70.12.5 Riak

Riak is a set of scalable distributed NoSQL databases developed by Basho Technologies. Riak KV is a key-value database with time-to-live feature so that older data is deleted automatically [90]. It can be queried through secondary indexes, search via Apache Solr, and MapReduce. Riak TS is designed for time-series data. It co-locates related data on the same physical cluster for faster access [91]. Riak S2 is designed to store large objects like media files and software binaries [92]. The databases are available in both open source and commercial versions with multiclustereplication provided only in later. REST APIs are available for these databases.

-- check content --

70.12.6 ZHT

ZHT can be described as “ a zero-hop distributed hash table” [618]. Distributed hash tables effectively break a hash table up and assign different nodes responsibility for managing different pieces of the larger hash table [742]. To retrieve a value in a distributed hash table, one needs to find the node that is responsible for the managing the key value pair of interest [742]. In general, every node that is a part of the distributed hash table has a reference to the closest two nodes in

the node list [742]. In a ZHT, however, every node contains information concerning the location of every other node [389]. Through this approach, ZHT aims to provide “high availability, good fault tolerance, high throughput, and low latencies, at extreme scales of millions of nodes” [389]. Some of the defining characteristics of ZHT are that it is light-weight, allows nodes to join and leave dynamically, and utilizes replication to obtain fault tolerance among others [389].

-- check content --

70.12.7 Berkeley DB

Berkeley DB is a family of open source, NoSQL key-value database libraries [95]. It provides a simple function-call API for data access and management over a number of programming languages, including C, C++, Java, Perl, Tcl, Python, and PHP. Berkeley DB is embedded because it links directly into the application and runs in the same address space as the application [94]. As a result, no inter-process communication, either over the network or between processes on the same machine, is required for database operations. It is also extremely portable and scalable, it can manage databases up to 256 terabytes in size.

For data management, Berkeley DB offers advanced services, such as concurrency for many users, ACID transactions, and recovery [478].

Berkeley DB is used in a wide variety of products and a large number of projects, including gateways from Cisco, Web applications at Amazon.com and open-source projects such as Apache and Linux.

-- check content --

70.12.8 Kyoto/Tokyo Cabinet – check content –

Tokyo Cabinet and Kyoto Cabinet are libraries of routines for managing a database [649] [382]. The database normally is a simple data file containing records having a key value pair structure. Every key and value is serial bytes with variable length. Both binary data and character string can be used as a key and a value. There is no concept of data tables nor data types like RDBMS or DBMS. Records are organized in hash table, B+ tree, or fixed-length array. Tokyo and Kyoto cabinets both are developed as a successor of GDBM and QDBM which are library routines for managing database as well. Tokyo Cabinet is written in the C language, and is provided as API of C, Perl, Ruby, Java, and Lua. Tokyo Cabinet is available on platforms which have API conforming to C99 and POSIX. Whereas Kyoto Cabinet is written in the C++ language, and is provided as API of C++, C, Java, Python, Ruby, Perl, and Lua. Kyoto Cabinet is available on platforms which have API conforming to C++03 with the TR1 library extensions. Both are free software licenced under GNU (General Public Licence). Kyoto Cabinet is more powerful and has convenient library structure than Tokyo and recommends people to use Kyoto [649]. Since they use key-value pair concept, you can store a record with a key and a value, delete a record using the key and even retrieve a record using the key. Both have smaller size of database file, faster processing speed and provide effective backup procedures.

-- check content --

70.12.9 Tycoon

Tycoon/ Kyoto Tycoon is a lightweight database server developed by FLL labs and is a distributed Key-value store [609] [385]. It is very useful in handling cache data persistent data of various

applications. Kyoto Tycoon is also a package of network interface to the DBM called Kyoto Cabinet which contains a library of routines for managing a database [384]. Tycoon is composed of a server process that manages multiple databases. This renders high concurrency enabling it to handle more than 10 thousand connections at the same time.

-- check content --

70.12.10 Tyrant

Tyrant provides network interfaces to the database management system called Tokyo Cabinet. Tyrant is also called as Tokyo Tyrant. Tyrant is implemented in C and it provides APIs for Perl, Ruby and C. Tyrant provides high performance and concurrent access to Tokyo Cabinet. The results of performance experiments between Tyrant and Memcached + MySQL can be viewed in the blog [666].

Tyrant was written and maintained by FAL Labs [665]. However, according to FAL Labs, their latest product Kyoto Tycoon is more powerful and convenient server than Tokyo Tyrant [664].

-- check content --

70.12.11 MongoDB

MongoDB is a NoSQL database which uses collections and documents to store data as opposed to the relational database where data is stored in tables and rows. In MongoDB a collection is a container for documents, whereas a document contains key-value pairs for storing data. As MongoDB is a NoSQL database, it supports dynamic schema design allowing documents to have different fields. The database uses a document storage and data interchange format called BSON, which provides a binary representation of JSON-like documents.

MongoDB provides high data availability by way of replication and sharding. High cost involved in data replication can be reduced by horizontal data scaling by way of shards where data is scattered across multiple servers. It reduces query cost as the query load is distributed across servers. This means that both read and write performance can be increased by adding more shards to a cluster. Which document resides on which shard is determined by the shard key of each collection.

As far as data backup and restore is concerned the default MongoDB storage engines natively support backup of complete data. For incremental backups one can use MongoRocks that is a third party tool developed by Facebook.

70.12.12 Espresso

Espresso is a document-oriented distributed data serving platform that plays an important role in LinkedIn's central data pipeline [328]. It currently powers approximately 30 LinkedIn applications including Member Profile, InMail, etc and also hosts some of its most important member data. Espresso provides a hierarchical data model in which the databases and table schema are defined in JSON. Some of the key components of Espresso include: (1) Router: which is a stateless HTTP Proxy and also acts as an entry point for all client requests in Espresso. The Router uses local cached routing table to manage the partition among all the storage nodes within the cluster. (2) Storage Node: are the building blocks of the storage and each one of them hosts a set of partition. (3) Helix: is responsible for cluster management in Espresso. (4) Databus: are responsible for capturing change to transport source transactions in commit order.

All the above mentioned components together enable Espresso to achieve real-time secondary

indexing, on-the-fly schema evolution and also a timeline consistent change capture stream.

-- check content --

70.12.13 CouchDB – check content –

The Apache Software Foundation makes CouchDB available as an option for those seeking an open-source, NoSQL, document-oriented database. CouchDB, or cluster of unreliable commodity hardware database, stores data as a JSON-formatted document [387]. Documents can consist of a variety of field types, e.g., text, booleans or lists, as well as metadata used by the software. CouchDB does not limit the number of fields per document, and it does not require any two documents to consist of matching or even similar fields [60]. That is, the document has structure, but the structure can vary by document. CouchDB coordinates cluster activities using the master-master mode by default, which means it does not have any one in charge of the cluster. However, a cluster can be set up to write all data to single node, which is then replicated across the cluster. Either way, the system can only offer eventual consistency. CouchDB serves as the basis of Couchbase, Inc’s Couchbase Server [150].

-- check content --

70.12.14 Couchbase Server

Couchbase, Inc. offers Couchbase Server (CBS) to the marketplace as a NoSQL, document-oriented database alternative to traditional relationship- oriented database management systems as well as other NoSQL competitors. The basic storage unit, a *document*, is a “data structure defined as a collection of named fields”. The document utilizes JSON, thereby allowing each document to have its own individual schema [273].

CBS combines the in-memory capabilities of Membase with CouchDB’s inherent data store reliability and data persistency. Membase functions in RAM only, providing the highest-possible speed capabilities to end users. However, Membase’s in-ram existence limits the amount of data it can use. More importantly, it provides no mechanism for data recovery if the server crashes. Combining Membase with CouchDB provides a persistent data source, mitigating the disadvantages of either product. In addition, CouchDB + membase allows the data size “to grow beyond the size of RAM” [110].

CBS is written in Erlang/OTP, but generally shortened to just Erlang. In actuality, it is written in “Erlang using components of OTP alongside some C/C++”, It runs on an Erlang virtual machine known as BEAM [725] [193].

Out-of-the-box benefits of Erlang/OTP include dynamic type setting, pattern matching and, most importantly, actor-model concurrency. As a result, Erlang code virtually eliminates the possibility of inadvertent deadlock scenarios. In addition, Erlang/OTP processes are lightweight, spawning new processes does not consume many resources and message passing between processes is fast since they run in the same memory space. Finally, OTP’s process supervision tree makes Erlang/OTP extremely fault-tolerant. Error handling is indistinguishable from a process startup, easing testing and bug detection [565].

CouchDB’s design adds another layer of reliability to CBS. CouchDB operates in *append-only* mode, so it adds user changes to the tail of database. This setup resists data corruption while taking a snapshot, even if the server continues to run during the procedure [360].

Finally, CB uses the Apache 2.0 License, one of several open-source license alternatives [584].

-- check content --

70.12.15 IBM Cloudant

Cloudant is based on both Apache-backed CouchDB project and the open source BigCouch project. IBM Cloudant is an open source non-relational, distributed database service as service (DBaaS) that provides integrated data management, search and analytics engine designed for web applications. Cloudant's distributed service is used the same way as standalone CouchDB, with the added advantage of data being redundantly distributed over multiple machines [732].

70.12.16 Pivotal Gemfire

A real-time, consistent access to data-intensive applications is provided by a open source, data management platform named Pivotal Gemfire. "GemFire pools memory, CPU, network resources, and optionally local disk across multiple processes to manage application objects and behavior". The main features of Gemfire are high scalability, continuous availability, shared nothing disk persistence, heterogeneous data sharing and parallelized application behavior on data stores to name a few. In Gemfire, clients can subscribe to receive notifications to execute their task based on a specific change in data. This is achieved through the continuous querying feature which enables event-driven architecture. The shared nothing architecture of Gemfire suggests that each node is self-sufficient and independent, which means that if the disk or caches in one node fail the remaining nodes remaining untouched. Additionally, the support for multi-site configurations enable the user to scale horizontally between different distributed systems spread over a wide geographical network [6].

70.12.17 HBase

Apache Hbase is a distributed column-oriented database which is built on top of HDFS (Hadoop Distributed File System). HBase is an open source, versioned, distributed, non-relational database modelled after Google's Bigtable [67]. Similar to Bigtable providing harnessing distributed file storage system offered by Google file system, Apache Hbase provides similar capabilities on top of Hadoop and HDFS. Moreover, Hbase supports random, real-time CRUD (Create/Read/Update/Delete) operations.

Hbase is a type of NoSQL database and is classified as a key value store. In HBase, value is identified with a key where both of them are stored as byte arrays. Values are stored in the order of keys. HBase is a database system where the tables have no schema. Some of the companies that use HBase as their core program are Facebook, Twitter, Adobe, Netflix etc.

-- check content --

70.12.18 Google Bigtable

Google Bigtable is a NoSQL database service, built upon several Google technologies, including Google File System, Chubby Lock Service, and SSTable [251]. Designed for Big Data, Bigtable provides high performance and low latency and scales to hundreds of petabytes [251]. Bigtable powers many core Google products, such as Search, Analytics, Maps, Earth, Gmail, and YouTube. Bigtable also drives Google Cloud Datastore and influenced Spanner, a distributed NewSQL database also developed by Google [712] [719]. Since May 6, 2015, Bigtable has been available to the public as Cloud Bigtable [719].

70.12.19 LevelDB

LevelDB is a light-weight, single-purpose library for persistence with bindings to many platforms [388]. It is a simple open source on-disk key/value data store built by Google, inspired by BigTable and is used in Google Chrome and many other products. It supports arbitrary byte arrays as both keys and values, singular get, put and delete operations, batched put and delete, bi-directional iterators and simple compression using the very fast Snappy algorithm. It is hosted on GitHub under the New BSD License and has been ported to a variety of Unix-based systems, Mac OS X, Windows, and Android. It is not an SQL database and does not support SQL queries. Also, it has no support for indexes. Applications use LevelDB as a library, as it does not provide a server or command-line interface.

-- check content --

70.12.20 Megastore and Spanner

Spanner is Google's distributed database which is used for managing all google services like play, gmail, photos, picasa, app engine etc Spanner is distributed database which spans across multiple clusters, datacenters and geo locations [145]. Spanner is structured in such a way so as to provide non blocking reads, lock free transactions and atomic schema modification. This is unlike other noSql databases which follow the CAP theory i.e. you can choose any two of the three: Consistency, Availability and Partition-tolerance. However, spanner gives an edge by satisfying all three of these. It gives you atomicity and consistency along with availability, partition tolerance and synchronized replication. Megastore bridges the gaps found in google's bigtable. As google realized that it is difficult to use bigtable where the application requires constantly changing schema. Megastore offers a solution in terms of semi-relational data model. Megastore also provides a transactional database which can scale unlike relational data stores and synchronous replication [108]. Replication in megastore is supported using Paxos. Megastore also provides versioning. However, megastore has a poor write performance and lack of a SQL like query language. Spanners basically adds what was missing in Bigtable and megastore. As a global distributed database spanner provides replication and globally consistent reads and writes. Spanner deployment is called universe which is a collections of zones. These zones are managed by singleton universe master and placement driver. Replication in spanner is supported by Paxos state machine. Spanner was put into evaluation in early 2011 as F1 backend (F1 is Google's advertisement system) which was replacement to mysql. Overall spanner fulfills the needs of relational database along with scaling of noSQL database. All these features make google run all their apps seamlessly on spanner infrastructure.

-- check content --

70.12.21 Accumulo

Apache Accumulo, a highly scalable structured store based on Google's BigTable, is a sorted, distributed key/value store that provides robust, scalable data storage and retrieval. Accumulo is written in Java and operates over the Hadoop Distributed File System (HDFS), which is part of the popular Apache Hadoop project. Accumulo supports efficient storage and retrieval of structured data, including queries for ranges, and provides support for using Accumulo tables as input and output for MapReduce jobs. Accumulo features automatic load-balancing and partitioning, data compression and fine-grained security labels. Much of the work Accumulo does involves maintaining certain properties of the data, such as organization, availability, and integrity, across many commodity-class machines [38].

70.12.22 Cassandra

Apache Cassandra is an open-source distributed database management for handling large volume of data across commodity servers [47]. It works on asynchronous masterless replication technique leading to low latency and high availability. It is a hybrid between a key-value and column oriented database. A table in cassandra can be viewed as a multi dimensional map indexed by a key. It has its own “Cassandra Query language (CQL)” query language for data extraction and mining. One of the demerits of such structure is it does not support joins or subqueries. It is a java based system which can be administered by any JMX compliant tools.

-- check content --

70.12.23 RYA

Rya is a “scalable system for storing and retrieving RDF data in a cluster of nodes” [516]. RDF stands for Resource Description Framework [516]. RDF is a model that facilitates the exchange of data on a network [276]. RDF utilizes a form commonly referred to as a triple, an object that consists of a subject, predicate, and object [516]. These triples are used to describe resources on the Internet [516]. Through new storage and querying techniques, Rya aims to make accessing RDF data fast and easy [551].

-- check content --

70.12.24 Sqrl

-- check content --

70.12.25 Neo4J

Neo4J is a popular ACID compliant graph database management system developed by Neo technology [704]. In this database everything is stored as nodes or edges, both of which can be labeled. Labels help in narrowing and simplifying the search process through the database [534]. It is a highly scalable software and can be distributed across multiple machines. The graph query language that accompanies the software has traversal framework which makes it fast and powerful [442]. The Neo4J is often used for clustering. It offers two feature clustering solutions: Causal Clustering and Highly available clustering [441]. Casual clustering focuses on safety, scalability and causal consistency in the graph [443]. The highly available cluster places importance to fault tolerance as each instance in the cluster has full copies of data in their local database.

-- check content --

70.12.26 graphdb

A Graph Database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data [727]. The Graph is a concept which directly relates the data items in the store. The data which is present in the store is linked together directly with the help of relationships. It can be retrieved with a single operation. Graph database allow simple and rapid retrieval of complex hierarchical structures that are difficult to model in relational systems.

There are different underlying storage mechanisms used by graph databases. Some graphdb depend on a relational engine and store the graph data in a table, while others use a key-value store or document-oriented database for storage. Thus, they are inherently called as NoSQL structures. Data

retrieval in a graph database requires a different query language other than SQL. Some of the query languages used to retrieve data from a graph database are Gremlin, SPARQL, and Cypher. Graph databases are based on graph theory. They employ the concepts of nodes, edges and properties.

-- check content --

70.12.27 Yarcdata – check content –

Yarcdata is Cray subsidiary providing Analytics products, namely the Urika Agile Analytics Platform and Graph Engine. Cray’s Urika (Universal RDF Integration Knowledge Appliance) system is a hardware platform designed specifically to provide high-speed graph-retrieval for relationship analytics [152]. Urika is a massively parallel, multi-threaded, shared-memory computing device designed to store and retrieve massive graph datasets. The system can import and host massive heterogeneous graphs represented in the resource description framework (RDF) format and can retrieve descriptive graph patterns specified in a SPARQL query.

Urika-GD is a big data appliance for graph analytics helps enterprises gain key insights by discovering relationships in big data [153]. Its highly scalable, real-time graph analytics warehouse supports ad hoc queries, pattern-based searches, inferencing and deduction. The Urika-GD appliance complements an existing data warehouse or Hadoop cluster by offloading graph workloads and interoperating within the existing analytics workflow

Cray Graph Engine is a semantic database using Resource Description Framework (RDF) triples to represent the data, SPARQL as the query language and extensions to support mathematical algorithms [547].

The paper “Graph mining meets the semantic web” outlines the implementation of graph mining algorithms using SPARQL [557].

-- check content --

70.12.28 AllegroGraph

“AllegroGraph is a database technology that enables businesses to extract sophisticated decision insights and predictive analytics from their highly complex, distributed data that can’t be answered with conventional databases, i.e., it turns complex data into actionable business insights” [19]. It can be viewed as a closed source database that is used for storage and retrieval of data in the form of triples (triple is a data entity composed of subject-predicate-object like “Professor teaches students”). Information in a triple store is retrieved using a query language. Query languages can be classified into database query languages or information retrieval query languages. The difference is that a database query language gives exact answers to exact questions, while an information retrieval query language finds documents containing requested information. Triple format represents information in a machine-readable format. Every part of the triple is individually addressable via unique URLs - for example, the statement “Professor teaches students” might be represented in RDF (Resource Description Framework). Using this representation, semantic data can be queried [20].

-- check content --

70.12.29 Blazegraph

Blazegraph is a graph database also supporting property graph, capable of clustered deployment. A graph database is a NoSQL database. It is based on a graph theory of nodes and edges where each

node represents an element such as user or business and each edge represents relationship between two nodes. It is mainly used for storing and analyzing data where maintaining interconnections is essential. Data pertaining to social media is best example where graph database can be used.

Blazegraph's main focus is large scale complex graph analytics and query. The Blazegraph database runs on graphics processing units (GPU) to speed graph traversals. [300]

Lets now see how Blazegraph handles data. [611] **Blazegraph data can be accessed** using REST APIs.

Blazegraph supports Apache TinkerPop, which is a graph computing framework.

For graph data mining, Blazegraph implements GAS (Gather, Apply, Scatter) model as a service.

-- check content --

70.12.30 Facebook Tao

In the paper published in USENIX annual technical conference, Facebook Inc describes TAO (The Association and Objects) as [109] a geographically distributed data store that provides timely access to the social graph for Facebook's demanding workload using a fixed set of queries. It is deployed at Facebook for many data types that fit its model. The system runs on thousands of machines, is widely distributed, and provides access to many petabytes of data. TAO represents social data items as Objects (user) and relationship between them as Associations (liked by, friend of). TAO cleanly separates the caching tiers from the persistent data store allowing each of them to be scaled independently. To any user of the system it presents a single unified API that makes the entire system appear like 1 giant graph database [672].

-- check content --

70.12.31 Titan:db

Titan:db is a distributed graph database that can support of thousands of concurrent users interacting with a single massive graph database that is distributed over the clusters [648]. It is open source with liberal Apache 2 license. Its main components are storage backend, search backend, and TinkerPop graph stack. Titan provides support for various storage backends and also linear scalability for a growing data and user base. It inherits features such as 'Gremlin' query language and 'Rexter' graph server from TinkerPop [647]. For huge graphs, Titan uses a component called Titan-hadoop which compiles Gremlin queries to Hadoop MapReduce jobs and runs them on the clusters. Titan is basically optimal for smaller graphs.

-- check content --

70.12.32 Jena

Jena is an open source Java Framework provided by Apache for semantic web applications ([679]). It provides a programmatic environment for RDF, RDFS and OWL, SPARQL, GRDDL, and includes a rule-based inference engine. Semantic web data differs from conventional web applications in that it supports a web of data instead of the classic web of documents format. The presence of a rule based inference engine enable Jena to perform a reasoning based on OWL and RDFS ontologies [653]. ' The architecture of Jena contains three layers: Graph layer, model layer and Ontology layer. The graph layer forms the base for the architecture. It does not have an extensive RDF implementation and serves more as a Service provider Interface. It provides

classes/methods that could be further extended [653]. The model layer extends the graph layer and provides objects of type 'resource' instead of 'node' to work with. The ontology layer enables one to work with triples.

-- check content --

70.12.33 Sesame

Sesame is framework which can be used for the analysis of RDF (Resource Description Framework) data. Resource Description Framework (RDF) is a model that facilitates the interchange of data on the Web [12]. Using RFD enables us to merge data even if the underlying schemas differ. Sesame has now officially been integrated into RDF4J Eclipse project [13]. Sesame takes in the natively written code as the input and then performs a series of transformations, generating kernels for various platforms. In order to achieve this, it makes use of the feature identifier, impact predictor, source-to-source translator and the auto-tuner [347]. The feature identifier is concerned with the extraction and detection of the architectural features that are important for application performance. The impact predictor determines the performance impact of the core features extracted above. A source-to-source translator transforms the input code into a parametrized one; while the auto-tuner helps find the optimal solution for the processor.

-- check content --

70.12.34 Public Cloud: Azure Table

FORMAT WRONG

Microsoft offers its NoSQL Azure Table product to the market as a low-cost, fast and scalable data storage option [583]. Table stores data as collections of key-value combinations, which it terms *properties*. Table refers to a collection of properties as an *entity*. Each entity can contain a mix of properties. The mix of properties can vary between each entity, although each entity may consist of no more than 255 properties [640].

Although data in Azure Table will be structured via key-value pairs, Table provides just one mechanism for the user to define relationships between entities: the entity's *primary key*. The primary key, which Microsoft sometimes calls a *clustered index*, consists of a PartitionKey and a RowKey. The PartitionKey indicates the group, a.k.a partition, to which the user assigned the entity. The RowKey indicates the entity's relative position in the group. Table sorts in ascending order by the PartitionKey first, then by the RowKey using lexical comparisons. As a result, numeric sorting requires fixed-length, zero-padded strings. For instance, Table sorts *111* before *2*, but will sort *111* after *002* [417].

Azure Table is considered best-suited for infrequently accessed data storage.

-- check content --

70.12.35 Amazon Dynamo

Amazon explains DynamoDB as a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale [www.dyndb]. It is a fully managed cloud database and supports both document and key-value store models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad tech, IoT, and many other applications. DynamoDB can be easily integrated with big-data processing tools like Hadoop. It can also be integrated with AWS Lambda, an event driven platform, which enables

creating applications that can automatically react to data changes. At present there are certain limits to DynamoDB. Amazon has listed all the limits in a web page titled *Limits in DynamoDB* <http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html>

-- check content --

70.12.36 Google DataStore

Google Cloud Datastore is a NoSQL document database built for automatic scaling, high performance, and ease of application development [247]. Though Cloud Datastore interface has many of the features similar to traditional databases, but as a NoSQL database, it differs from the SQL in the way as it describes relationships between various data objects. It also provides a number of features that relational databases are not optimally suited to provide, including high-performance at a very large scale and high-reliability. The Google Cloud DataStore can have different kinds of properties for the same kind of entities, unlike the Relational Database where they are represented in rows. For example, the difference between entities can have the properties with the same name but having different values. The flexible schema maps naturally to object-oriented and scripting languages.

Non-relational databases have become popular recently, especially for web applications that require high-scalability and performance with high-availability. Non-relational databases such as Cloud DataStore let developers to choose an optimal balance between strong consistency and eventual consistency for each application. This allows developers to combine the benefits of both the database structures [266]. Datastore is designed to automatically scale to very large data sets, allowing applications to maintain high performance as they receive more traffic. Datastore also provides a number of features that relational databases are not optimally suited to provide, including high-performance at a very large scale and high-reliability [247].

70.13 File management

70.13.1 iRODS

The Integrated Rule-Oriented Data System (iRODS) is open source data management software. iRODS is released as a production-level distribution aimed at deployment in mission critical environments. It virtualizes data storage resources, so users can take control of their data, regardless of where and on what device the data is stored. The development infrastructure supports exhaustive testing on supported platforms. The plugin architecture supports microservices, storage systems, authentication, networking, databases, rule engines, and an extensible API [334]. iRODS implements data virtualization, allowing access to distributed storage assets under a unified namespace, and freeing organizations from getting locked in to single-vendor storage solutions. iRODS enables data discovery using a metadata catalog that describes every file, every directory, and every storage resource in the iRODS Zone. iRODS automates data workflows, with a rule engine that permits any action to be initiated by any trigger on any server or client in the Zone. iRODS enables secure collaboration, so users only need to log in to their home Zone to access data hosted on a remote Zone [335].

-- check content --

70.13.2 NetCDF

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array oriented scientific data. NetCDF was developed

and is maintained at Unidata, part of the University Corporation for Atmospheric Research (UCAR) Community Programs (UCP). Unidata is funded primarily by the National Science Foundation [182] [444]. The purpose of the Network Common Data Form (netCDF) interface is to support the creation, efficient access, and sharing of data in a form that is self-describing, portable, compact, extendible, and archivable. Version 3 of netCDF is widely used in atmospheric and ocean sciences due to its simplicity. NetCDF version 4 has been designed to address limitations of netCDF version 3 while preserving useful forms of compatibility with existing application software and data archives [182]. NetCDF consists of: (a) A conceptual data model (b) A set of binary data formats (c) A set of APIs for C/Fortran/Java

70.13.3 CDF

Common Data Format is a conceptual data abstraction for storing, manipulating, and accessing multidimensional data sets [438]. CDF differs from traditional physical file formats by defining form and function as opposed to a specification of the bits and bytes in an actual physical format.

CDF's integrated dataset is composed by following two categories: (a) Data Objects - scalars, vectors, and n-dimensional arrays. (b) Metadata - set of attributes describing the CDF in global terms or specifically for a single variable [437].

The self-describing property (metadata) allows CDF to be a generic, data-independent format that can store data from a wide variety of disciplines. Hence, the application developer remains insulated from the actual physical file format for reasons of conceptual simplicity, device independence, and future expandability. CDF data sets are portable on any of the CDF-supported platforms and accessible with CDF applications or layered tools. To ensure the data integrity in a CDF file, checksum method using MD5 algorithm is employed [167].

Compared to HDF format, CDF permitted cross-linking data from different instruments and spacecraft in ISTP with one development effort [729]. CDF is widely supported by commercial and open source data analysis/visualization software such as IDL, MATLAB, and IBM's Data Explorer (XP).

-- check content --

70.13.4 HDF

-- check content --

70.13.5 OPeNDAP

-- check content --

70.13.6 FITS

FITS stand for 'Flexible Image Transport System'. It is a standard data format used in astronomy. FITS data format is endorsed by NASA and International Astronomical Union. FITS can be used for transport, analysis and archival storage of scientific datasets and support multi-dimensional arrays, tables and headers sections [207]. FITS is actively used and developed [208]. FITS can be used for digitization of contents like books and magazines. Vatican Library uses FITS for long term preservation of their book, manuscripts and other collection [205]. Matlab, a language used for technical computing supports fits [206]. A 2011 paper explains how to perform processing of

astronomical images on Hadoop using FITS [359].

-- check content --

70.13.7 RCFile

RCFile (Record Columnar File) is a big data placement data structure that supports fast data loading and query processing coupled with efficient storage space utilization and adaptive to dynamic workload environments [536]. It is designed for data warehousing systems that uses map-reduce. The data is stored as a flat file comprising of binary key/value pairs. The rows are partitioned first and then the columns are partitioned in each row and the respective meta-data for each row is stored in the key part for that row and the values comprises of the data part of the row. Storing the data in this format enables RCFile to accomplish fast loading and query processing. A shell utility is available for reading RCFile data and metadata [535]. RCFile has been chosen in Facebook data warehouse system as the default option [290]. It has also been adopted by Hive and Pig, the two most widely used data analysis systems developed in Facebook and Yahoo!

-- check content --

70.13.8 ORC

ORC files were created as part of the initiative to massively speed up Apache Hive and improve the storage efficiency of data stored in Apache Hadoop. ORC is a self-describing type-aware columnar file format designed for Hadoop workloads. It is optimized for large streaming reads, but with integrated support for finding required rows quickly. Storing data in a columnar format lets the reader read, decompress, and process only the values that are required for the current query. Because ORC files are type-aware, the writer chooses the most appropriate encoding for the type and builds an internal index as the file is written. ORC files are divided in to stripes that are roughly 64MB by default. The stripes in a file are independent of each other and form the natural unit of distributed work. Within each stripe, the columns are separated from each other so the reader can read just the columns that are required [89].

70.13.9 Parquet

Apache parquet is the column Oriented data store for Apache Hadoop ecosystem and available in any data processing framework, data model or programming language [216]. It stores data such that the values in each column are physically stored in contiguous memory locations. As it has the columnar storage, it provides efficient data compression and encoding schemes which saves storage space as the queries that fetch specific column values need not read the entire row data and thus improving performance. It can be implemented using the Apache Thrift framework which increases its flexibility to work with a number of programming languages like C++, Java, Python, PHP, etc.

70.14 Data Transport

70.14.1 BitTorrent

Bittorrent is P2P communication protocol commonly used for sending and receiving the large digital files like movies and audioclips. In order to upload and download file, user have to download bittorrent client which implement the bittorrent protocol. Bittorrent uses the principle of swarming

and tracking [104]. It divides the files in large number of chunk and as soon as file is received it can be server to the other users for downloading. So rather than downloading one entire large file from one source, user can download small chunk from the different sources of linked users in swarm. Bittorrent trackers keeps list of files available for transfer and helps the swarm user find each other.

Using the protocol, machine with less configuration can serve as server for distributing the files. It result in increase in the downloading speed and reduction in origin server configuration.

Few popular bittorrent client in μ Torrent, qBittorrent.

-- check content --

70.14.2 HTTP

70.14.3 FTP

FTP is an acronym for File Transfer Protocol [224]. It is network protocol standard used for transferring files between two computer systems or between a client and a server. It is part of the Application layer of the Internet Protocol Suite and works along with HTTP/SSH. It follows a client-server model architecture. Secure systems asks the client to authenticate themselves using a Username and Password registered with the server to access the files via FTP. The specification for FTP was first written by Abhay Bhushan in 1971 and is termed as RFC114 [www-rfc114]. The current specification, RFC959 in use was written in 1985. Several other versions of the specification are available which provides firewall friendly FTP access, additional security extensions, support for IPV6 and passive mode file access respectively. FTP can be used in command line in most of the operating systems to transfer files. There are FTP clients such as WinSCP, FileZilla etc. which provides a graphical user interface to the clients to authenticate themselves (sign on) and access the files from the server.

-- check content --

70.14.4 SSH

SSH is a cryptographic network protocol to provide a secure channel between two clients over an unsecured network [600]. It uses public-key cryptography for authenticating the remote machine and the user. The public-private key pairs could be generated automatically to encrypt the network connection. ssh-keygen utility could be used to generate the keys manually. The public key then could be placed on the all the computers to which the access is required by the owner of the private key. SSH runs on the client-server model where a server listens for incoming ssh connection requests. It's generally used for remote login and command execution. It's other important uses include tunneling (required in cloud computing) and file transfer (SFTP). OpenSSH is an open source implementation of network utilities based on SSH [601].

-- check content --

70.14.5 Globus Online (GridFTP)

GridFTP is a enhancement on the File Transfer Protocol (FTP) which provides high-performance, secure and reliable data transfer for high-bandwidth wide-area networks. The most widely used implementation of GridFTP is Globus Online [274]. GridFTP achieves efficient use of bandwidth by using multiple simultaneous TCP streams. Files can be downloaded in pieces simultaneously from multiple sources; or even in separate parallel streams from the same source. GridFTP allows

transfers to be restarted automatically and handles network unavailability with a fault tolerant implementation of FTP. The underlying TCP connection in FTP has numerous settings such as window size and buffer size. GridFTP allows automatic (or manual) negotiation of these settings to provide optimal transfer speeds and reliability.

-- check content --

70.14.6 Flume

Flume is distributed, reliable and available service for efficiently collecting, aggregating and moving large amounts of log data [626]. Flume was created to allow you to flow data from a source into your Hadoop environment. In Flume, the entities you work with are called sources, decorators, and sinks. A source can be any data source, and Flume has many predefined source adapters. A sink is the target of a specific operation. A decorator is an operation on the stream that can transform the stream in some manner, which could be to compress or uncompress data, modify data by adding or removing pieces of information, and more [310].

70.14.7 Sqoop

Apache Sqoop is a tool to transfer large amounts of data between Apache Hadoop and sql databases [624]. The name is a Portmanteau of SQL + Hadoop. It is a command line interface application which supports incremental loads of complete tables, free form (custom) SQL Queries and allows the use of saved and scheduled jobs to import latest updates made since the last import. The imports can also be used to populate tables in Hive or Hbase. Sqoop has the option of export, which allows data to be transferred from Hadoop into a relational database. Sqoop is supported in many different business integration suits like Informatica Big Data Management, Pentaho Data Integration, Microsoft BI Suite and Couchbase [697].

70.14.8 Pivotal GPLOAD/GPFDIST

Greenplum Database is a shared nothing, massively parallel processing solution built to support next generation data warehousing and Big Data analytics processing [248]. In its new distribution under Pivotal, Greenplum Database is called Pivotal (Greenplum) Database.

gpfdist is Greenplum's parallel file distribution program [270]. It is used by readable external tables and gpload to serve external table files to all Greenplum Database segments in parallel. It is used by writable external tables to accept output streams from Greenplum Database segments in parallel and write them out to a file.

gpload is data loading utility is used to load data into Greenplum's external table in parallel [248].

Google has an invention relating to integrating map-reduce processing techniques into a distributed relational database. An embodiment of the invention is implemented by Greenplum as gpfdist [139].

-- check content --

70.15 Cluster Resource Management

70.15.1 Mesos

Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and

run effectively v

citewww-mesos. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elasticsearch) with API's for resource management and scheduling across entire datacenter and cloud environments.

The resource scheduler of Mesos supports a generalization of max-min fairness, termed Dominant Resource Fairness (DRF) scheduling discipline, which allows to harmonize execution of heterogeneous workloads (in terms of resource demand) by maximizing the share of any resource allocated to a specific framework [7] [240].

Mesos uses containers for resource isolation between processes. In the context of Mesos, the two most important resource-isolation methods to know about are the control groups (cgroups) built into the Linux kernel, and Docker. The difference between using hyper-V, Docker containers, cgroup is described in detail in the book "Mesos in action" [319].

-- check content --

70.15.2 Yarn

Yarn (Yet Another Resource Negotiator) is Apache Hadoop's cluster management project [135]. It's a resource management technology which make a pace between, the way applications use Hadoop system resources & node manager agents. Yarn, "split up the functionalities of resource management and job scheduling/monitoring". The NodeManager watch the resource (cpu, memory, disk, network) usage the container and report the same to ResourceManager. Resource manager will take a decision on allocation of resources to the applications. ApplicationMaster is a library specific to application, which requests/negotiate resources from ResourceManager and launch and monitoring the task with NodeManager (s) [602]. ResourceManager have two majors: Scheduler and ApplicationManager. Scheduler have a task to schedule the resources required by the application. ApplicationManger holds the record of application who require resource. It validates (whether to allocate the resource or not) the application's resource requirement and ensure that no other application already have register for the same resource requirement. Also it keeps the track of release of resource [280].

-- check content --

70.15.3 Helix

Helix is a data management system getting developed by IBM which helps the users to do explicatory analysis of the data received from various sources following different formats. This system would help organize the data by providing links between data collected across various sources despite of the knowledge of the data sources schemas. It also aims at providing the data really required for the user by extracting the important information from the data. This would plan to target the issue by maintaining the "knowledge base of schemas" and "context-dependent dynamic linkage", The system can get the schema details either from the knowledge base being maintained or can even get the schema from the data being received. As the number of users for helix increases the linkages gets stronger and would provide better data quality [197].

-- check content --

70.15.4 Llama

Llama stands for leveraging learning to automatically manage algorithms. There has been a phenomenal improvement in algorithm portfolio and selection approaches. The main drawback of them is that their implementation is specific to a problem domain and customized which leads to the difficulty of exploring new techniques for certain problem domains. Llama has been developed to provide an extensible toolkit which can initiate exploration of a variety of portfolio techniques over a wide range of problem domains. It is modular and implemented as an R package. It leverages the extensive library of machine learning algorithms and techniques in R [375]. Llama can be regarded as a framework which provides the prerequisites for initiating automatic portfolio selectors. It provides a set of methods for combining several trivial approaches of portfolio selection into sophisticated techniques. The primary reason behind the introduction of Llama was to help the researchers working in algorithm selection, algorithm portfolios, etc. and can be just used as a tool for designing the systems [375].

70.15.5 Google Omega

-- check content --

70.15.6 Facebook Corona

Corona is a new scheduling framework developed by facebook which separates the cluster resource management from job coordination. Facebook, employed the MapReduce implementation from Apache Hadoop since 2011 for job scheduling. The scheduling MapReduce framework has its limitations with the scalability as when the number of jobs at facebook grew in the next few years. Another limitation of Hadoop was it was a pull-based scheduling model as the task tracker have to provide a heartbeat to the job tracker to indicate that it is running which associated with a pre-defined delay, that was problematic for small jobs [198]. Hadoop MapReduce is also constrained by its static slot-based resource management model where a MapReduce cluster is divided into a fixed number of map and reduce slots based on a static configurations so the slots are not utilized completely anytime the cluster workload does not fit the static configuration.

Corona improves over the Hadoop MapReduce by introducing a cluster manager whose only purpose is to track the nodes in the cluster and the amount free resources [198]. A dedicated job tracker is created for each job and can run either in the same process as the client (for small jobs) or as a separate process in the cluster (for large jobs). The other difference is that it uses a push-based scheduling whose implementation does not involve a periodic heartbeat and thus scheduling latency is minimized. The cluster manager also implements a fair-share scheduling as it has access to the full snapshot of the cluster for making the scheduling decisions. Corona is used as an integral part of the Facebook's data infrastructure and is helping power big data analytics for teams across the company.

70.15.7 Celery

“Celery is an asynchronous task queue/job queue based on distributed message passing. The focus of celery is mostly on real-time operation, but it equally scheduling. In celery there are execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing, Eventlet, or gevent. Tasks can execute asynchronously (in the background) or synchronously (wait until ready). Celery is easy to integrate with web framework. Celery is written in python whereas the protocol can be implemented in any language” [118]. “Celery is a simple,

flexible, and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system“[119].

-- check content --

70.15.8 HTCondor

HTCondor is a specialized workload management system for compute-intensive jobs. HTCondor provides various features like (a) job queuing mechanism, (b) scheduling policy, (c) resource monitoring, (d) priority scheme and (e) resource management just as other full-featured batch systems. “Users submit their serial or parallel jobs to HTCondor, HTCondor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion”. HTCondor can be used to manage a cluster of dedicated compute nodes. HTCondor uses unique mechanisms to harness wasted CPU power from idle desktop workstations. “The ClassAd mechanism in HTCondor provides an extremely flexible and expressive framework for matching resource requests (jobs) with resource offers (machines). Jobs can easily state both job requirements and job preferences”. “HTCondor incorporates many of the emerging Grid and Cloud-based computing methodologies and protocols” [689].

-- check content --

70.15.9 SGE

Sun Grid Engine (SGE) renamed to Oracle Grid Engine (OGE) is a grid computing cluster software system [475]. Grid Engine is a high performance computing cluster used for managing job queuing in distributed and parallel environment. It can accept, schedule, dispatch and manage the execution of single, parallel user jobs in a remote or distributed manner. It also manages the resource allocation to those jobs. The resources can be anything like processors, storage, RAM and licenses for softwares. The latest stable release of OGE is termed as 6.2u8 which came out in October 1, 2012.

OGE supports a vast array of features like: Topology-aware scheduling and thread binding, advanced fault tolerance mechanisms for job scheduling, web interface based status reporting and ability to use different scheduling algorithms, etc. OGE runs on several platforms including AIX, BSD, Linux, Solaris, OS X, Tru64, Windows, etc. It is under deployment phase for IBM's 64-bit operating system z/OS. Standard Grid cluster comprises of one master host and many execution hosts. There is an option of creating shadow master hosts which would take the master's place in case of a system crash. Notable deployments of OGE include: TSUBAME supercomputer at the Tokyo Institute of Technology, Ranger at the Texas Advanced Computing Center (TACC) and San Diego Supercomputer Center (SDSC).

-- check content --

70.15.10 OpenPBS

Portable Batch System (or simply PBS) is the name of computer software that performs job scheduling. Its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources. It is often used in conjunction with UNIX cluster environments [698]. OpenPBS is the original open source version of PBS. There are more commercialized versions of the same software. One of the key features of OpenPBS is that it supports millions of cores with fast job dispatch and minimal latency. It meets unique site goals and SLAs by balancing job

turnaround time and utilization with optimal job placement. OpenPBS also includes automatic fail-over architecture with no single point of failure - jobs are never lost, and jobs continue to run despite failures. It is built upon a Flexible Plugin Framework which simplifies administration with enhanced visibility and extensibility [489].

70.15.11 Moab

Moab HPC Suite is a workload management and resource orchestration platform that automates the scheduling, managing, monitoring, and reporting of HPC workloads on massive scale. It uses multi-dimensional policies and advanced future modeling to optimize workload start and run times on diverse resources. It integrates and accelerates the workloads management across independent clusters by adding grid-optimized job submission. Moab's unique intelligent and predictive capabilities evaluate the impact of future orchestration decisions across diverse workload domains (HPC, HTC, Big Data, and Cloud VMs)[422].

70.15.12 Slurm

Simple Linux Utility for Resource Management (SLURM) workload manager is an open source, scalable cluster resource management tool used for job scheduling in small to large Linux cluster using multi-core architecture. SLURM has three key functions. First, it allocates resources to users for some duration with exclusive and/or non-exclusive access. Second, it enables users to start, execute and monitor jobs on the resources allocated to them. Finally, it intermediates to resolve conflicts on resources for pending work by maintaining them in a queue [562]. The slurm architecture has following components: a centralized manager to monitor resources and work, may have a backup manager, daemon on each server to provide fault-tolerant communications, an optional daemon for clusters with multiple managers and tools to initiate, terminate and report about jobs in a graphical view with network topology. It also provides around twenty additional plugins that could be used for functionalities like accounting, advanced reservation, gang scheduling, back fill scheduling and multifactor job prioritization. Though originally developed for Linux, SLURM also provides full support on platforms like AIX, FreeBSD, NetBSD and Solaris [563] [582].

-- check content --

70.15.13 Torque

-- check content --

70.15.14 Globus Tools – check content –

The Globus Toolkit is an open source toolkit organized as a collection of loosely coupled components [589]. These components consist of services, programming libraries and development tools designed for building Grid-based applications. GT components fall into five broad domain areas: Security, Data Management, Execution Management, Information Services, and Common Runtime [214]. These components enable a broader *Globus ecosystem* of tools and components that build on or interoperate with GT functionality to provide a wide range of useful application-level functions [246]. Since 2000, companies like Fujitsu, IBM, NEC and Oracle have pursued Grid strategies based on the Globus Toolkit [246].

-- check content --

70.15.15 Pilot Jobs

In pilot job, an application acquires a resource so that it can be delegated some work directly by the application; instead of requiring some job scheduler. The issue of using a job scheduler is that a waiting queue is required. Few examples of Pilot Jobs are the [527] Falkon lightweight framework and [371] HTCaaS. Pilot jobs are typically associated with both Parallel computing as well as Distributed computing. Their main aim is to reduce the dependency on queues and the associated multiple wait times.

Using pilot jobs enables us to have a multilevel technique for the execution of various workloads. This is so because the jobs are typically acquired by a placeholder job and they relayed to the workloads [655].

70.16 File systems

70.16.1 HDFS

Hadoop provides distributed file system framework that uses Map reduce (Distributed computation framework) for transformation and analyses of large dataset. Its main work is to partition the data and other computational tasks to be performed on that data across several clusters. HDFS is the component for distributed file system in Hadoop. An HDFS cluster primarily consists of a Name Node and Data Nodes. Name Node manages the file system metadata such as access permission, modification time, location of data and Data Nodes store the actual data. When user applications or Hadoop frameworks request access to a file in HDFS, Name Node service responds with the Data Node locations for the respective individual data blocks that constitute the whole of the requested file [289].

70.16.2 Swift

-- check content --

70.16.3 Haystack

Haystack is an open source project working with data from internet of Things, aim to standardise the semantic data model generated from smart devices, homes, factories etc. It include automation, control, energy, HVAC, lighting and other environmental systems [287].

Building block of Project haystack is on TagModel tagging of metadata stored in key/value pair applied to entity such id, dis, sites, geoAddr, tz. Structure the primary structure of haystack is based on three entities, Site location of single unit, equip physical or logical piece of equipment within site, point sensor, actuator or setpoint value for equip, it also includes weather outside weather condition. TimeZone time series data is most important factor it is foundation for sensor and operational data. Captured data not always associated with measurable unit, however it provides facility to associate the data points. Commonly Supported units like Misc, Area, Currency, Energy, Power, Temperature, Temperature differential, Time, Volumetric Flow. The data often represented in 2D tabular form for tagged entities. It supports the query language for filtering over the data, data exposed through REST API in JSON format.

-- check content --

70.16.4 f4

As the amount of data Facebook stores continues to increase, the need for quick access and efficient storage of data continues to rise. Facebook stores a class of data in Binary Large Objects (BLOBs), which can be created once, read many times, never modified, and sometimes deleted. Haystack, Facebook's traditional BLOB storage system is becoming increasingly inefficient. The storage efficiency is measured in the effective-replication-factor of BLOBs.

f4 BLOB storage system provides an effective-replication-factor lower than that of Haystack. f4 is simple, modular, scalable, and fault tolerant. f4 currently stores over 65PBs of logical BLOBs, with a reduced effective-replication-factor from 3.6 to either 2.8 or 2.1 [433].

70.16.5 Cinder

“Cinder is a block storage service for Openstack” [126]. Openstack Compute uses ephemeral disks meaning that they exist only for the life of the Openstack instance i.e. when the instance is terminated the disks disappear. Block storage system is a type of persistent storage that can be used to persist data beyond the life of the instance. Cinder provides users with access to persistent block-level storage devices. It is designed such that users can create block storage devices on demand and attach them to any running instances of OpenStack Compute [526]. This is achieved through the use of either a reference implementation (LVM) or plugin drivers for other storage. Cinder virtualizes the management of block storage devices and provides end users with a self-service API to request and consume those resources without requiring any knowledge of where their storage is actually deployed or on what type of device [126].

70.16.6 Ceph

Ceph is open-source storage platform providing highly scalable object, block as well as file-based storage. Ceph is a unified, distributed storage system designed for excellent performance, reliability and scalability [544]. Ceph Storage clusters are designed to run using an algorithm called CRUSH (Controlled Replication Under Scalable Hashing) which replicates and re-balance data within the cluster dynamically to ensure even data distribution across cluster and quick data retrieval without any centralized bottlenecks.

Ceph's foundation is the Reliable Autonomic Distributed Object Store (RADOS), which provides applications with object, block, and file system storage in a single unified storage cluster - making Ceph flexible, highly reliable and easy to manage [546]. Ceph decouples data and metadata operations by eliminating file allocation tables and replacing them with generating functions which allows RADOS to leverage intelligent OSDs to manage data replication, failure detection and recovery, low-level disk allocation, scheduling, and data migration without encumbering any central server (s) [683].

The Ceph Filesystem is a POSIX-compliant filesystem that uses a Ceph Storage Cluster to store its data [545]. Ceph's dynamic subtree partitioning is a uniquely scalable approach, offering both efficiency and the ability to adapt to varying workloads. Ceph Object Storage supports two compatible interfaces: Amazon S3 and Openstack Swift.

-- check content --

70.16.7 FUSE

FUSE (Filesystem in Userspace) ‘‘is an interface for userspace programs to export a filesystem to the Linux kernel’’ [225]. The FUSE project consists of two components: the fuse kernel module and the libfuse userspace library. libfuse provides the reference implementation for communicating with the FUSE kernel module. The code for FUSE itself is in the kernel, but the filesystem is in userspace. As per a 2006 paper on HPTFS which has been built on top of FUSE [772]. It mounts a tape as normal file system based data storage and provides file system interfaces directly to the application. Another implementation of FUSE FS is CloudBB [767]. Unlike conventional filesystems CloudBB creates an on-demand two-level hierarchical storage system and caches popular files to accelerate I/O performance. On evaluating performance of real data-intensive HPC applications in Amazon EC2/S3, results show CloudBB improves performance by up to 28.7 times while reducing cost by up to 94.7% compared to the ones without CloudBB.

Some more implementation examples of FUSE are - mp3fs (A VFS to convert FLAC files to MP3 files instantly), Copy-FUSE (To access cloud storage on Copy.com), mtpfs (To mount MTP devices) etc.

-- check content --

70.16.8 Gluster

-- check content --

70.16.9 Lustre

The Lustre file system is an open-source, parallel file system that supports many requirements of leadership class HPC simulation environments and Enterprise environments worldwide [471]. Because Lustre file systems have high performance capabilities and open licensing, it is often used in supercomputers. Lustre file systems are scalable and can be part of multiple computer clusters with tens of thousands of client nodes, tens of petabytes of storage on hundreds of servers, and more than a terabyte per second of aggregate I/O throughput. Lustre file systems a popular choice for businesses with large data centers, including those in industries such as meteorology, simulation, oil and gas, life science, rich media, and finance. Lustre provides a POSIX compliant interface and many of the largest and most powerful supercomputers on Earth today are powered by the Lustre file system.

-- check content --

70.16.10 IBM Spectrum Scale, formerly GPFS

General Parallel File System (GPFS) was rebranded as IBM Spectrum Scale on February 17, 2015 [733].

Spectrum Scale is a clustered file system, developed by IBM, designed for high performance. It ‘‘provides concurrent high-speed file access to applications executing on multiple nodes of clusters’’ and can be deployed in either shared-nothing or shared disk modes [733]. Spectrum Scale is available on AIX, Linux, Windows Server, and IBM System Cluster 1350 [733]. Due to its focus on performance and scalability, Spectrum Scale has been utilized in compute clusters, big data and analytics - including support for Hadoop Distributed File System (HDFS), backups and restores, and private clouds [309].

-- check content --

70.16.11 GFFS

The Global Federated File System (GFFS) is a computing technology that allows linking of data from Windows, Mac OS X, Linux, AFS, and Lustre file systems into a global namespace, making them available to multiple systems [503]. It is a federated, secure, standardized, scalable, and transparent mechanism to access and share resources across organizational boundaries. It is useful when, for data resources, boundaries do not require application modification and do not disrupt existing data access patterns. It uses FUSE to handle access control and allows research collaborators on remote systems to access a shared file system. Existing applications can access resources anywhere in the GFFS without modification. It helps in rapid development of code, which can then be exported via GFFS and implemented in-place on a given computational resource or Science Gateway.

-- check content --

70.16.12 Public Cloud: Amazon S3

Amazon Simple Storage Service (Amazon S3) is storage object which provides a simple web service interface to store and retrieve any amount of data from anywhere on the web [26]. With Amazon S3, users can store as much data as they want and can scale it up and down based on the requirements. For developers Amazon S3 provides full REST API's and SDK's which can be integrated with third-party technologies. Amazon S3 is also deeply integrated with other AWS services to make it easier to build solutions that use a range of AWS services which include Amazon CloudFront, Amazon CloudWatch, Amazon Kinesis, Amazon RDS, Amazon Glacier etc. Amazon S3 provides automatic encryption of data once the data is uploaded in the cloud. Amazon S3 uses the concept of Buckets and Objects for storing data wherein Buckets are used to store objects. Amazon S3 services can be used using the Amazon Console Management [670]. The steps for using the Amazon S3 are as follows: (1) Sign up for Amazon S3 (2) After sign up, create a Bucket in your account, (3) Create an object which might be a file or folder, and (4) Perform operations on the object which is stored in the cloud.

-- check content --

70.16.13 Azure Blob

Azure Blob storage is a service that stores unstructured data in the cloud as objects/blobs. Blob storage can store any type of text or binary data, such as a document, media file, or application installer. Blob storage is also referred to as object storage. The word 'Blob' expands to Binary Large Object [33]. There are three types of blobs in the service offered by Windows Azure namely block, append and page blobs [239]. 1. Block blobs are a collection of individual blocks with unique block ID. The block blobs allow the users to upload large amounts of data. 2. Append blobs are optimized blocks that help in making the operations efficient. 3. Page blobs are a compilation of pages. They allow random read and write operations. While creating a blob, if the type is not specified they are set to block type by default. All the blobs must be inside a container in your storage. Azure Blob storage is a service for storing large amounts of unstructured object data, such as text or binary data, that can be accessed from anywhere in the world via HTTP or HTTPS. You can use Blob storage to expose data publicly to the world, or to store application data privately. Common uses of Blob storage include serving images or documents directly to a browser, storing

files for distributed access, streaming video and audio, storing data for backup and restore, disaster recovery, and archiving and storing data for analysis by an on-premises or Azure-hosted service. Azure Storage is massively scalable and elastic with an auto-partitioning system that automatically load-balances your data. Blob storage is a specialized storage account for storing your unstructured data as blobs (objects) in Azure Storage. Blob storage is similar to existing general-purpose storage accounts and shares all the great durability, availability, scalability, and performance features. Blob storage has two types of access tiers that can be specified, hot access tier, which will be accessed more frequently, and a cool access tier, which will be less frequently accessed. There are many reasons why you should consider using BLOB storage. Perhaps you want to share files with clients, or off-load some of the static content from your web servers to reduce the load on them [33].

-- check content --

70.16.14 Google Cloud Storage

Google Cloud Storage is the cloud enabled storage offered by Google [263]. It is unified object storage. To have high availability and performance among different regions in the geo-redundant storage offering. If you want high availability and redundancy with a single region one can go for *Regional* storage. *Nearline* and *Coldline* are the different archival storage techniques. *Nearline* storage offering is for the archived data which the user access less than once a month. *Coldline* storage is the storage which is used for the data which is touched less than once a year.

All the data in Google Cloud storage belongs inside a project. A project will contains different buckets. Each bucket has different objects. We need to make sure that the name of the bucket is unique across all Google cloud name space. And the name of the objects should unique in a bucket.

-- check content --

70.17 Interoperability

70.17.1 Cloudmesh

Cloudmesh client allows to easily manage virtual machines, containers, HPC tasks, through a convenient client and API. Hence cloudmesh is not only a multi-cloud, but a multi-hpc environment that allows also to use container technologies. Cloudmesh is currently developed as part of classes taught at Indiana University.

-- check content --

70.17.2 Libvirt

Libvirt is an open source API to manage hardware virtualization developed by Red Hat. It is a standard C library but has accessibility from other languages such as Python, Perl, Java and others [393]. Multiple virtual machine monitors (VMM) or hypervisors are supported such as KVM, QEMU, Xen, Virtuozzo, VMWare ESX, LXC, and BHyve. It can be divided into five categories such as hypervisor connection, domain, network, storage volume and pool [352]. It is accessible by many operating systems such as Linux, FreeBSD, Mac OS, and Windows OS.

-- check content --

70.17.3 Libcloud

[392] Libcloud is a python library that allows to interact with several popular cloud service providers. It is primarily designed to ease development of software products that work with one or more cloud services supported by Libcloud. It provides a unified API to interact with these different cloud services. Current API includes methods for list, reboot, create, destroy, list images and list sizes. [391] lists Libcloud key component APIs Compute, Storage, Load Balancers, DNS, Container and Backup. Compute API allows users to manage cloud servers. Storage API allows users to manage cloud object storage and also provides CDN management functionality. Load balancer, DNS and Backup API's allows users to manage their respective functionalities, as services, and related products of different cloud service providers. Container API allows users to deploy containers on to container virtualization platforms. Libcloud supports Python 2, Python 3 and PyPy.

-- check content --

70.17.4 JClouds

The primary goals of cross-platform cloud APIs is that application built using these APIs can be seamlessly ported to different cloud providers citecloud-portability-book. The APIs also bring interoperability such that cloud platforms can communicate and exchange information using these common or shared interfaces. Jclouds or apache jclouds is a java based library to provide seamless access to cloud platforms [344]. Jclouds library provides interfaces for most of cloud providers like docker, openstack, amazon web services, microsoft azure, google cloud engine etc. It will allow users build applications which can be portable across different cloud environments. Key components of jcloud are:

1. Views: abstracts functionality from a specific vendor and allow user to write more generic code. For example odbc abstracts the underlying relational data source. However, odbc driver converts to native format. In this case user can switch databases without rewriting the application. Jcloud provide following views: blob store, compute service, loadBalancer service
2. API: APIs are requests to execute a particular functionality. Jcloud provide a single set of APIs for all cloud vendors which is also location aware. If a cloud vendor doesn't support customers from a particular region the API will not work from that region.
3. Provider: a particular cloud vendor is a provider. Jcloud uses provider information to initialize its context.
4. Context: it can be termed as a handle to a particular provider. Its like a ODBC connection object. Once connection is initialized for a particular database, it can used to make any api call.

Jclouds provides test library to mock context, APIs etc to different providers so that user can write unit test for his implementation rather than waiting to test with the cloud provider. Jcloud library certifies support after testing the interfaces with live cloud provider. These features make jclouds robust and adoptable, hiding most of the complexity of cloud providers.

-- check content --

70.17.5 TOSCA

70.17.6 OCCI

The Open Cloud Computing Interface (OCCI) is a RESTful Protocol and API that provides specifications and remote management for the development of *interoperable tools* [753]. It supports IaaS, PaaS and SaaS and focuses on integration, portability, interoperability, innovation and extensibility. It provides a set of documents that describe an OCCI Core model, contain best practices of interaction with the model, combined into OCCI Protocols, explain methods of communication between components via HTTP protocol introduced in the OCCI Renderings, and define infrastructure for IaaS presented in the OCCI Extensions.

The current version 1.2 OCCI consists of seven documents that identify required and optional components. Of the Core Model. In particular, the following components are required to implement: (a) Core Model, (b) HTTP protocol, (c) Text rendering and (d) JSON rendering. Meanwhile, Infrastructure, Platform and SLA models are optional. The OCCI Core model defines instance types and

provides a layer of abstraction that allows the OCCI client to interact with the model without knowing of its potential structural changes. The model supports extensibility via inheritance and using mixin types that represent ability to add new components and capabilities at run-time [754].

The OCCI Protocol defines the common set of names provided for the IaaS cloud services user that specify requested system requirements. It is often denoted as *resource templates* or *flavours* [757].

OCCI RESTful HTTP Protocol describes communications between server and client on OCCI platform via HTTP protocol [755]. It defines a minimum set of HTTP headers and status codes to ensure compliance with the OCCI Protocol. Separate requirements for Server and Client for versioning need to be implemented using HTTP Server header and User-Agent header respectively.

JSON rendering protocol provides JSON specifications to allow “render OCCI instances independently of the protocol being used.” In addition, it provides details of the JSON object declaration, OCCI Action Invocation, object members required for OCCI Link Instance Rendering, “location maps to OCCI Core’s source and target model attributes and kind maps to OCCI Core’s target” to satisfy OCCI Link Instance Source/Target Rendering requirements [756]. Finally, it specifies various attributes and collection rendering requirements. The text rendering process is deprecated and will be removed from the next major version [758].

-- check content --

70.17.7 CDMI

The Storage Networking Industry Association (SNIA) is a non-profit organization formed by various companies, suppliers and consumers of data storage and network products [77]. SNIA defines various standards to ensure the quality and interoperability of various storage systems. One of the standards defined by SNIA for providers and users of cloud is Cloud Data Management Interface (CDMI). According latest issue of CDMI, “CDMI International Standard is intended for application developers who are implementing or using cloud storage [78]. It documents how to access cloud storage and to manage the data stored there.” It defines functional interface for applications that will use cloud for various functionalities like create, retrieve, update and delete data elements from the cloud. These interface could be used to manage containers along with the data. The interface could be used by administrative and management applications as well. Also, the CDMI specification uses RESTful principles in the interface design. All the standards issued

on CDMI can be found on SNIA web page [79].

-- check content --

70.17.8 Whirr

Apache Whirr is a set of libraries for running cloud services, which provides a cloud-neutral way to run services [692]. This is achieved by using cloud-neutral provisioning and storage libraries such as jclouds and libcloud. Whirr's API should be built on top these libraries and is not exposed to the users. It is also a common service API, in which the details of its working are, particular to the service. Whirr provides smart defaults for services by which any properly configured system can run quickly, while still being able to override settings as needed. Whirr can also be used as a command line tool for deploying clusters. It uses low level API libraries to work with providers which was mentioned in the [693].

70.17.9 Saga

SAGA (Simple API for Grid Applications) provides an abstraction layer to make it easier for applications to utilize and exploit infra effectively. With infrastructure being changed continuously its becoming difficult for most applications to utilize the advances in hardware. SAGA API provides a high level abstraction of the most common Grid functions so as to be independent of the diverse and dynamic Grid environments [346]. This shall address the problem of applications developers developing an application tailored to a specific set of infrastructure. SAGA allows computer scientists to write their applications at high level just once and not to worry about low level hardware changes. SAGA provides this high level interface which has the underlying mechanisms and adapters to make the appropriate calls in an intelligent fashion so that it can work on any underlying grid system. "SAGA was built to provide a standardized, common interface across various grid middleware systems and their versions" [250].

As SAGA is to be implemented on different types of middleware it does not specify a single security model but provides hooks to interfaces of various security models. The SAGA API provides a set of packages to implement its objectivity: SAGA supports data management, resource discovery, asynchronous notification, event generation, event delivery etc. It does so by providing set of functional packages namely SAGA file package, replica package, stream package, RPC package, etc. SAGA provides interoperability by allowing the same application code to run on multiple grids and also communicate with applications running on others [346].

70.17.10 Genesis

70.18 DevOps

70.18.1 Docker (Machine, Swarm)

Docker is an open-source container-based technology. A container allows a developer to package up an application and all its part including the stack it runs on, dependencies it is associated with and everything the application requires to run within an isolated environment. Docker separates Application from the underlying Operating System in a similar way as Virtual Machines separates the Operating System from the underlying hardware. Dockerizing an application is lightweight in comparison with running the application on the Virtual Machine as all the containers share the same underlying kernel, the Host OS should be same as the container OS (eliminating guest OS) and an average machine cannot have more than few VMs running o them.

Docker Machine is a tool that lets you install Docker Engine on virtual hosts, and manage the hosts with docker-machine commands [406]. You can use Machine to create Docker hosts on your local Mac or Windows machine, on your company network, in your data center, or on cloud providers like AWS or Digital Ocean. For Docker 1.12 or higher swarm mode is integrated with the Docker Engine, but on the older versions with Machine's swarm option, user can configure a swarm cluster. Docker Swarm provides native clustering capabilities to turn a group of Docker engines into a single, virtual Docker Engine. "With these pooled resources user can scale out your application as if it were running on a single, huge computer" [171]. Docker Swarm can be scaled up to 1000 Nodes or up to 50,000 containers

70.18.2 Puppet

Puppet is an open source software configuration management tool [518]. This aims at automatic configuration of the software applications and infrastructure. This configuration is done using the easy to use language. Puppet works on major linux distributions and also on microsoft windows, it is also cross-platform application making it easy to manage and portable [517].

Puppet works with a client server model. All the clients (nodes) which needs to be managed will have 'Puppet Agent' installed and 'Puppet Master' contains the configuration for different hosts this demon process rund on master server. The connection between 'Puppet Master' and 'Puppet agent' will be established using the secured SSL connection. The configuration at client will be validated as per the set up in Puppet master at a predefined interval. If configuration at client is not matching with the master puppet agent fetches the changes from master [297].

Puppet is developed by Puppet Labs using ruby language and released as GNU General Public License (GPL) until version 2.7.0 and the Apache License 2.0 after that [518].

-- check content --

70.18.3 Chef

Chef is a configuration management tool. It is implemented in Ruby and Erlang. Chef can be used to configure and maintain servers on-premise as well as cloud platforms like Amazon EC2, Google Cloud Platform and Open Stack. Explanation of concept called 'recipes', in Chef, to manage server applications and utilities such as database servers like MySQL, or HTTP servers like Apache HTTP and systems like Apache Hadoop [404].

Chef is available in open source version and it also has commercial products for the companies which need it [122].

-- check content --

70.18.4 Ansible

Ansible is an IT automation tool that automates cloud provisioning, configuration management, and application deployment [543]. Once Ansible gets installed on a control node, which is an agentless architecture, it connects to a managed node through the default OpenSSH connection type [714].

As with most configuration management softwares, Ansible distinguishes two types of servers: controlling machines and nodes. First, there is a single controlling machine which is where orchestration begins. Nodes are managed by a controlling machine over SSH. The controlling machine describes the location of nodes through its inventory.

Ansible manages machines in an agent-less manner. Ansible is decentralized, if needed, Ansible can easily connect with Kerberos, LDAP, and other centralized authentication management systems.

-- check content --

70.18.5 SaltStack

SaltStack (also Salt) platform is a Python-based open-source configuration management software and remote execution engine, which makes systems and configuration management software for the orchestration and automation of CloudOps, ITOps and DevOps at scale [554]. SaltStack is used to manage all the data center things including any cloud, infrastructure, virtualization, application stack, software or code. Salt is built on two major concepts -- remote execution and configuration management [435]. In the remote execution system, Salt leverages Python to accomplish complex tasks with single-function calls. The configuration management system in Salt, called States, builds upon the remote execution foundation to create repeatable, enforceable configuration for the minions (connects to the master and treats the master as the source)

-- check content --

70.18.6 Boto

The latest version of Boto is Boto3 [234]. Boto3 is the Amazon Web Services (AWS) Development Kit (SDK) for Python [237]. It enables the Python developers to make use of services like Amazon S3 and Amazon EC2 [236]. It provides object oriented APIs along with low-level direct service [235]. It provides simple in-built functions and interfaces to work with Amazon S3 and EC2.

Boto3 has two distinct levels of APIs - client and resource [236]. One-to-one mappings to underlying HTTP API is provided by the client APIs. Resource APIs provide resource objects and collections to perform various actions by accessing the attributes. Boto3 also comes with 'waiters'. Waiters are used for polling status changes in AWS, automatically. Boto3 has these waiters for both the APIs - client as well as resource.

70.18.7 Cobbler

Cobbler is a Linux provisioning system that facilitates and automates the network based system installation of multiple computer operating systems from a central point using services such as DHCP, TFTP and DNS [423]. It is a nifty piece of code that assembles all the usual setup bits required for a large network installation like TFTP, DNS, PXE installation trees and automates the process. It can be configured for PXE, reinstallations and virtualized guests using Xen, KVM or VMware. Cobbler interacts with the koan program for re-installation and virtualization support. Cobbler builds the Kickstart mechanism and offers installation profiles that can be applied to one or many machines. Cobbler has features to dynamically change the information contained in a kickstart template (definition), either by passing variables called ksmeta or by using so-called snippets.

70.18.8 Xcat

xCAT is defined as extreme cloud/cluster administration toolkit. This open source software was developed by IBM and utilized on clusters based on either linux or a version of UNIX called AIX. With this service administrator is enabled with a number of capabilities including parallel system management, provision OS usage on virtual machines, and manage all systems remotely [762].

xCAT works with various cluster types such as high performance computing, horizontal scaling web farms, administrative, and operating systems. [313]

-- check content --

70.18.9 Razor

Razor is a hardware provisioning application, developed by Puppet Labs and EMC. Razor was introduced as open, pluggable, and programmable since most of the provisioning tools that existed were vendor-specific, monolithic, and closed. Razor can deploy both bare-metal and virtual systems. During boot the Razor client automatically discovers the inventory of the server hardware - CPUs, disk, memory, etc., feeds this to the Razor server in real-time and the latest state of every server is updated [519]. It maintains a set of rules to dynamically match the appropriate operating system images with server capabilities as expressed in metadata. User-created policy rules are referred to choose the preconfigured model to be applied to a new node. The node follows the model's directions, giving feedback to Razor as it completes various steps as specified in [520]. Models can include steps for handoff to a DevOps system or to any other system capable of controlling the node.

-- check content --

70.18.10 Juju

Juju (formerly Ensemble) is software from Canonical that provides open source service orchestration [93]. It is used to easily and quickly deploy and manage services on cloud and physical servers. Juju charms can be deployed on cloud services such as Amazon Web Services (AWS), Microsoft Azure and OpenStack. It can also be used on bare metal using MAAS. Approximately 300 charms available for services available in the Juju store [356]. Charms can be written in any language. It also supports Bundles which are pre-configured collection of Charms that helps in quick deployment of whole infrastructure.

-- check content --

70.18.11 Foreman

-- check content --

70.18.12 OpenStack Heat

Openstack Heat, a template deployment service was the project launched by Openstack, a cloud operating system similar to AWS Cloud Formation. Heat is an orchestration service which allows us to define resources over the cloud and connections amongst them using a simple text file called referred as a 'template' [399]. "A Heat template describes the infrastructure for a cloud application in a text file that is readable and writable by humans, and can be checked into version control" [292].

Once the execution environment has been setup and a user wants to modify the architecture of resources in the future, a user needs to simply change the template and check it in. Heat shall make the necessary changes. Heat provides 2 types of template - HOT (Heat Orchestration Template) and CFN (AWS Cloud Formation Template). The HOT can be defined as YAML and is not compatible with AWS. The CFN is expressed as JSON and follows the syntax of AWS Cloud Formation and

thus is AWS compatible. Further, heat provides an additional @parameters section in its template which can be used to parameterize resources to make the template generic.

-- check content --

70.18.13 Sahara

The Sahara product provides users with the capability to provision data processing frameworks (such as Hadoop, Spark and Storm) on OpenStack by specifying several parameters such as the version, cluster topology and hardware node details [472]. The solution allows for fast provisioning of data processing clusters on OpenStack for development and quality assurance and utilisation of unused computer power from a general purpose OpenStack IaaS Cloud [553]. Sahara is managed via a REST API with a User Interface available as part of OpenStack Dashboard.

-- check content --

70.18.14 Rocks

Rocks provides open cluster distribution solution is build targeting the scientist with less cluster experience to ease the process of deployment, managing, upgrading and scaling high performance parallel computing cluster [548]. It was initially build on linux however the latest version Rocks 6.2 Sidewinder is also available on CentOS. Rocks can help create a cluster in few days with default configuration and software packages. Rocks distribution package comes with high-performance distributed and parallel computing tools. It is used by NASA, the NSA, IBM Austin Research LAB, US Navy and many other institution for their projects.

-- check content --

70.18.15 Cisco Intelligent Automation for Cloud

Cisco Intelligent automation for cloud desires to help different service providers and software professionals in delivering highly secure infrastructure as a service on demand. It provides a foundation for organizational transformation by expanding the uses of cloud technology beyond its infrastructure [129]. From a single self-service portal, it automates standard business processes and sophisticated data center which is beyond the provision of virtual machines. Cisco Intelligent automation for cloud is a unified cloud platform that can deliver any type of service across mixed environments [418]. This leads to an increase in cloud penetration across different business and IT holdings. Its services range from underlying infrastructure to anything-as-a-service by allowing its users to evaluate, transform and deploy the IT and business services in a way they desire.

70.18.16 Ubuntu MaaS

-- check content --

70.18.17 Facebook Tupperware

Facebook Tupperware is a system which provisions services by taking requirements from engineers and mapping them to actual hardware allocations using containers [511]. Facebook Tupperware simplifies the task of configuring and running services in production and allows engineers to focus on actual application logic. The tupperware system consists of a Scheduler, Agent process and a Server Database. The Scheduler consists of set of machines with one of them as master and the

others in standby. The machines share state among them. The Agent process runs on each and every machine and manages all the tasks and co-ordinates with the Scheduler. The Server database stores the details of resources available across machines which is used by the scheduler for scheduling jobs and tasks. Tupperware allows for sandboxing of the tasks which allows for isolation of the tasks. Initially isolation was implemented using chroots but now it is switched to Linux Containers (LXC). The configuration for the container is done by a specific config file written in a dialect of python by the owner of the process.

70.18.18 AWS OpsWorks

AWS Opsworks is a configuration service provided by Amazon Web Services that uses Chef, a Ruby and Erlang based configuration management tool [720], to automate the configuration, deployment, and management of servers and applications. There are two versions of AWS Opsworks. The first, a fee based offering called AWS OpsWorks for Chef Automate, provides a Chef Server and suite of tools to enable full stack automation. The second, AWS OpsWorks Stacks, is a free offering in which applications are modeled as stacks containing various layers. Amazon Elastic Cloud Compute (EC2) instances or other resources can be deployed and configured in each layer of AWS OpsWorks Stacks [23].

70.18.19 OpenStack Ironic

Ironic project is developed and supported by OpenStack. Ironic provisions bare metal machines instead of virtual machines and functions as hypervisor API that is developed using open source technologies like Preboot Execution Environment (PXE), Dynamic Host Configuration Protocol (DHCP), Network Bootstrap Program (NBP), Trivial File Transfer Protocol (TFTP) and Intelligent Platform Management Interface (IPMI) [671]. A properly configured Bare Metal service with the Compute and Network services, could provision both virtual and physical machines through the Compute service's API. But, the number of instance actions are limited, due to physical servers and switch hardware. For example, live migration is not possible on a bare metal instance. The Ironic service has five key components. A RESTful API service, through which other components would interact with the bare metal servers, a Conductor service, various drivers, messaging queue and a database. Ironic could be integrated with other OpenStack projects like Identity (keystone), Compute (nova), Network (neutron), Image (glance) and Object (swift) services.

-- check content --

70.18.20 Google Kubernetes

Google Kubernetes is a cluster management platform developed by Google. Kubernetes is an open source system for “automating deployment, scaling and management of containerized applications” [377]. It primarily manages clusters through containers as they decouple applications from the host operating system dependencies and allowing their quick and seamless deployment, maintenance and scaling.

Kubernetes components are designed to extensible primarily through Kubernetes API. Kubernetes follows a master-slave architecture, Kubernetes Master controls and manages the clusters workload and communications of the system [703]. Its main components are etcd, API server, scheduler and controller manager. The individual Kubernetes nodes are the workers where containers are deployed. The components of a node are Kubelet, Kube-proxy and cAdvisor. Kubernetes makes it easier to run application on public and private clouds. It is also said to be self-healing due to

features like auto-restart and auto-scaling.

-- check content --

70.18.21 Buildstep

Buildsteps is an open software developed under MIT license. It is a base for Dockerfile and it activates Heroku-style application. Heroku is a platform-as-service (PaaS) that automates deployment of applications on the cloud. The program is pushed to the PaaS using git push, and then PaaS detects the programming language, builds, and runs application on a cloud platform [502]. Buildstep takes two parameters: a tar file that contains the application and a new application container name to create a new container for this application. Build script is dependent on buildpacks that are pre-requisites for buildstep to run. The builder script runs inside the new container. The resulting build app can be run with Docker using `docker build -t your_app_name` command. [249].

70.18.22 Gitreceive

Gitreceive is used to create an ssh+git user which can accept repository pushes right away and also triggers a hook script. Gitreceive is used to push code anywhere as well as extend your Git workflow. “Gitreceive dynamically creates bare repositories with a special pre-receive hook that triggers your own general gitreceive hook giving you easy access to the code that was pushed while still being able to send output back to the git user” Gitreceive can also be used to provide feedback to the user not only just to trigger code on git push. Gitreceive can used for the following: “(a) for putting a git push deploy interface in front of App Engine (b) Run your company build/test system as a separate remote (c) Integrate custom systems into your workflow (d) Build your own Heroku e)Push code anywhere” [750].

70.18.23 OpenTOSCA

The Topology and Orchestration Specification for Cloud Applications, TOSCA is a new standard facilitating platform independent description of Cloud applications. OpenTOSCA is a runtime for TOSCA-based Cloud applications. The runtime enables fully automated plan-based deployment and management of applications defined in the OASIS TOSCA packaging format CSAR, Cloud Service ARchive. The key tasks of OpenTOSCA, are to operate management operations, run plans, and manage state of the TOSCA [99].

70.18.24 Winery – check content –

Eclipse Winery is a “web-based environment to graphically model [Topology and Orchestration Specification for Cloud Applications] TOSCA topologies and plans managing these topologies” [221]. Winery is a “tool offering an HTML5-based environment for graph-based modeling of application topologies and defining reusable component and relationship types” [374]. This web-based interface enables users to drag and drop icons to create automated “provisioning, management, and termination of applications in a portable and interoperable way” [374]. Essentially, this web-based interface allows users to create an application topology, which “describes software and hardware components involved and relationships between them” as well a management plan, which “captures knowledge [regarding how] to deploy and manage an application” [374].

-- check content --

70.18.25 CloudML

CloudML a research project initiated by SINTEF in 2011 [578]. Cloud computing facilitates to shared and virtualized computer capabilities like storage, memory, CPU, GPU and networks, to user. There is multiple cloud provider, also the Iaas (Infrastructure-as-a-service) and Pass (Platform-as-a-service). To operate multiple cloud for applications, which requires multiple private, public, or hybrid clouds, limit the capability of each cloud solution. Solution provided by such cloud will gets incompatible with others. So, to providing the solution which can compatible with multi-cloud platform is a tedious job. To achieve this CloudML provides a “domain-specific modelling language along with run time environment” [578]. It provides the interoperability and provide vendor lock-in, also it provides the solution on specification of provisioning, deployment, and adaptation concerns of multi-cloud systems. At design time as well as runtime [578]. CloudML provides two level of abstraction while developing model for multi-cloud application: (a) Cloud Provider-Independent Model (CPIM), this specifies the provisioning and deployment. (b) Cloud Provider-Specific Model (CPSM), which filters the provisioning and deployment of multiple cloud application, according to its cloud. These two abstract approach help CloudML to achieve the multi-cloud application support [579].

70.18.26 Blueprints – check content –

In [307], it is explained that “IBM Blueprint has been replaced by IBM Blueworks Live.” In [306], IBM Blueworks Live is described “as a cloud-based business process modeller, belonging under the set of IBM SmartCloud applications” that as [308] states “drive[s] out inefficiencies and improve[s] business operations.” Similarly to Google Docs, IBM Blueworks Live is “designed to help organizations discover and document their business processes, business decisions and policies in a collaborative manner.” While Google Docs and IBM Blueworks Live are both simple to use in a collaborative manner, [306] explains that IBM Blueworks Live has the “capabilities to implement more complex models.”

70.18.27 Terraform – check content –

Terraform, developed by HashiCorp, is an infrastructure management tool, it has an open source platform as well as an enterprise version and uses infrastructure as a code to increase operator productivity. It’s latest release is Terraform 0.8 According to the website [622] it enables users to safely and predictably create, change and improve the production infrastructure and codifies APIs into declarative configuration files that can be shared amongst other users and can be treated as a code, edited, reviewed and versioned at the same time. The book [656] explains that it can manage the existing and popular service it provides as well as create customized in-house solutions. It builds an execution plan that describes what it can do next after it reaches a desired state to accomplish the goal state. It provides a declarative executive plan which is used for creating applications and implementing the infrastructures. Terraform is mainly used to manage cloud based and SaaS infrastructure, it also supports Docker and VMWare vSphere.

70.18.28 DevOpSlang

DevOpSlang serves as means of collaboration and provides the foundation to automate deployment and operations of an application. Technically, it is a domain specific language based on JavaScript Object Notation (JSON). JSON Schema is used to define a formal schema for DevOpSlang and complete JSON Schema definition of DevOpSlang is publicly available on GitHub project DevOpSlang: <http://github.com/jojow/devopslang> Devopsfiles are the technical artifacts (Unix

shell commands, Chef Scripts, etc.) rendered using DevOpsSlang to implement operations. Beside some meta data such as 'version' and 'author' Devopsfile defines operations like 'start' consisting of a single or multiple actions which specifies the command to run the application. Similarly, a 'build' operation can be defined to install the dependencies required to run the application. Different abstraction levels may be combined consistently such as a 'deploy' operation consisting of actions on the level of Unix shell commands and actions using portable Chef cookbooks [685].

70.18.29 Any2Api – check content –

This framework [686] allows user to wrap an executable program or scripts, for example scripts, chef cookbooks, ansible playbooks, juju charms, other compiled programs etc. to generate APIs from your existing code. These APIs are also containerized so that they can be hosted on a docker container, vagrant box etc Any2Api helps to deal with problems like scale of application, technical expertise, large codebase and different API formats. The generated API hide the tool specific details simplifying the integration and orchestration different kinds of artifacts. The APIfication framework contains various modules:

1. Invokers, which are capable of running a given type of executable for example cookbook invoker can be used to run Chef cookbooks
2. Scanners, which are capable of scanning modules of certain type for example cookbook scanner scans Chef cookbooks.
3. API impl generators, which are doing the actual work to generate the API implementation.

The final API implementation [684] is is packages with executable in container. The module is packaged as npm module. Currently any2api-cli provides a command line interface and web based interface is planned for future development. Any2Api is very useful for by devops to orchestrate open source ecosystem without dealing with low level details of chef cookbook or ansible playbook or puppet. It can also be very useful in writing microservices where services talk to each other using well defined APIs.

70.19 IaaS Management from HPC to hypervisors

70.19.1 Xen

Xen is the only open-source bare-metal hypervisor based on microkernel design [763]. The hypervisor runs at the highest privilege among all the processes on the host. It's responsibility is to manage CPU and memory and handle interrupts [764]. Virtual machines are deployed in the guest domain called DomU which has no access privilege to hardware. A special virtual machine is deployed in the control domain called Domain 0. It contains hardware drivers and the toolstack to control the VMs and is the first VM to be deployed. Xen supports both Paravirtualization and hardware assisted virtualization. The hypervisor itself has a very small footprint. It's being actively maintained by Linux Foundation under the trademark *XEN Project*. Some of the features included in the latest releases include em Reboot-free Live Patching (to enable application of security patches without rebooting the system) and KCONFIG support (compilation support to create a lighter version for requirements such as embedded systems) [765].

70.19.2 KVM

It is an acronym for Kernel-based Virtual Machine for the Linux Kernel that turns it into a hypervisor upon installation. It was originally developed by Qumranet in 2007 [379]. It has a kernel model and uses kernel as VMM. It only supports fully virtualized VMs. It is very active for Linux users due to its ease of use, it can be completely controlled by ourselves and there is an ease for migration from or to other platforms. It is built to run on a x86 machine on an Intel processor with virtualization technology extensions (VT-x) or an AMD-V. It supports 32 and 64 bit guests on a 64 bit host and hardware visualization features. The supported guest systems are Solaris, Linux, Windows and BSD Unix [380].

70.19.3 QEMU – check content –

QEMU (Quick Emulator) is a generic open source hosted hypervisor [731] that performs hardware virtualization (virtualization of computers as complete hardware platform, certain logical abstraction of their componentry or only the certain functionality required to run various operating systems) [735] and also emulates CPUs through dynamic binary translations and provides a set of device models, enabling it to run a variety of unmodified guest operating systems.

When used as an emulator, QEMU can run Operating Systems and programs made for one machine (ARM board) on a different machine (e.g. a personal computer) and achieve good performance by using dynamic translations. When used as a virtualizer, QEMU achieves near native performance by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using KVM kernel module in Linux [521].

Compared to other virtualization programs like VMWare and VirtualBox, QEMU does not provide a GUI interface to manage virtual machines nor does it provide a way to create persistent virtual machine with saved settings. All parameters to run virtual machine have to be specified on a command line at every launch. It's worth noting that there are several GUI front-ends for QEMU like virt-manager and gnome-box.

-- check content --

70.19.4 Hyper-V – check content –

Hyper-V is a native hypervisor which was first released alongside Windows Server 2008. It is available free of charge for all the Windows Server and some client operating systems since the release. Microsoft Hyper-V, is also codenamed as Viridian and formerly known as Windows Server Virtualization, is a native hypervisor. Xbox One also include Hyper-V, in which it would launch both Xbox OS and Windows 10. [730]

Hyper-V is used to create virtual machines on x86-64 systems which are running Windows. Windows 8 onwards, Hyper-V supersedes Windows Virtual PC as the hardware virtualization component of the client editions of Windows NT. A server computer running Hyper-V can be configured to expose individual virtual machines to one or more networks.

70.19.5 VirtualBox

-- check content --

70.19.6 OpenVZ – check content –

OpenVZ (Open Virtuozzo) is an operating system-level virtualization technology for Linux. It allows a physical server to run multiple isolated operating system instances, called containers, virtual private servers, or virtual environments (VEs). OpenVZ is similar to Solaris Containers and LXC. [474] While virtualization technologies like VMware and Xen provide full virtualization and can run multiple operating systems and different kernel versions, OpenVZ uses a single patched Linux kernel and therefore can run only Linux. All OpenVZ containers share the same architecture and kernel version. This can be a disadvantage in situations where guests require different kernel versions than that of the host. However, as it does not have the overhead of a true hypervisor, it is very fast and efficient. Memory allocation with OpenVZ is soft in that memory not used in one virtual environment can be used by others or for disk caching. [298] While old versions of OpenVZ used a common file system (where each virtual environment is just a directory of files that is isolated using chroot), current versions of OpenVZ allow each container to have its own file system. OpenVZ has four main features, [203] 1. OS virtualization: A container (CT) looks and behaves like a regular Linux system. It has standard startup scripts; software from vendors can run inside a container without OpenVZ-specific modifications or adjustment; A user can change any configuration file and install additional software; Containers are completely isolated from each other and are not bound to only one CPU and can use all available CPU power. 2. Network virtualization: Each CT has its own IP address and CTs are isolated from the other CTs meaning containers are protected from each other in the way that makes traffic snooping impossible; Firewalling may be used inside a CT 3. Resource management: All the CTs are use the same kernel. OpenVZ resource management consists of four main components: two-level disk quota, fair CPU scheduler, disk I/O scheduler, and user beancounters. 4. Checkpointing and live migration: Checkpointing allows to migrate a container from one physical server to another without a need to shutdown/restart a container. This feature makes possible scenarios such as upgrading your server without any need to reboot it: if your database needs more memory or CPU resources, you just buy a newer better server and live migrate your container to it, then increase its limits.

70.19.7 LXC – check content –

LXC (Linux Containers) is an operating-system-level virtualization method for running multiple isolated Linux systems (containers) on a control host using a single Linux kernel [396]. LXC are similar to the traditional virtual machines but instead of having separate kernel process for the guest operating system being run, containers would share the kernel process with the host operating system. This is made possible with the implementation of namespaces and cgroups. [695]

Containers are light weighed (As guest operating system loading and booting is eliminated) and more customizable compared to VM technologies. The basis for docker development is also LXC. [397]. Linux containers would work on the major distributions of linux this would not work on Microsoft Windows.

70.19.8 Linux-Vserver – check content –

Linux-VServers are used on web hosting services, pooling resources and containing any security breach. [509] “Linux servers consist of three building blocks Hardware, Kernel and Applications” the purpose of kernel is to provide abstraction layer between hardware and application. Linux-Vserver provides VPS securely partitioning the resources on computer system in such a way that process cannot mount denial of service out of the partition.

It utilises the power of Linux kernel and top of it with additional modification provides secure

layer to each process (VPS) feel like it is running separate system. By providing context separation, context capabilities, each partition called as security context, chroot barrier created on private directory of each VPS to prevent unauthorized modification. Booting VPS in new secure context is just matter of booting server, context is so robust to boot many server simultaneously.

The virtual servers shares same system calls, shares common file system, process within VS are queued to same scheduler that of host allowing guest process to run concurrently on SMP systems. No additional overhead of network virtualization. These few advantages of Linux-VServer.

70.19.9 OpenStack – check content –

OpenStack [588] is a free and open source cloud operating system mostly deployed as infrastructure as a service (IaaS) that allows us to control large pool of computers, storage, and networking resources. OpenStack is managed by OpenStack Foundation [215].

Just like cloud, OpenStack provides infrastructure which runs as platform upon which end users can create applications. Key components of OpenStack include: Nova: which is the primary computing engine, Swift: which is a storage system for object and files, Neutron: which ensures effective communication between each of the components of the OpenStack. Other components include: Cinder, Horizon, Keystone, Glance, Ceilometer and Heat. The main goal of Openstack is to allow business to build Amazon-like cloud services in their own data centers. OpenStack is licensed under the Apache 2.0 license [51]

70.19.10 OpenNebula – check content –

According to OpenNebula webpage [5] it provides simple but feature-rich and flexible solutions for the comprehensive management of virtualized data centers to enable private, public and hybrid IaaS clouds. It is a cloud computing platform for managing heterogeneous distributed data centers infrastructures. The OpenNebula toolkit includes features for management, scalability, security and accounting. It used in various sectors like hosting providers, telecom providers, telecom operators, IT service providers, supercomputing centers, research labs, and international research projects [469]. More about OpenNebula can be found in the following paper that is published at IEEE Computer Society [424]

70.19.11 Eucalyptus

Eucalyptus is a Linux-based open source software framework for cloud computing that implements Infrastructure as a Service (IaaS). IaaS are systems that give users the ability to run and control entire virtual machine instances deployed across a variety physical resources [453]. Eucalyptus is an acronym for “Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems.”

A Eucalyptus private cloud is deployed on an enterprise’s data center infrastructure and is accessed by users over the enterprise’s intranet. Sensitive data remains entirely secure from external interference behind the enterprise firewall [634].

70.19.12 Nimbus – check content –

Nimbus Infrastructure [448] is an open source IaaS implementation. It allows deployment of self-configured virtual clusters and it supports configuration of scheduling, networking leases, and usage metering.

Nimbus Platform [447] provides an integrated set of tools which enable users to launch large virtual clusters as well as launch and monitor the cloud apps. It also includes service that provides auto-scaling and high availability of resources deployed over multiple IaaS cloud. The Nimbus Platform tools are cloudinit.d, Phantom and Context Broker. In this paper [178], the use of Nimbus Phantom to deploy auto-scaling solution across multiple NSF FutureGrid clouds is explained. In this implementation Phantom was responsible for deploying instances across multiple clouds and monitoring those instance. Nimbus platform supports Nimbus, Open Stack, Amazon and several other clouds.

70.19.13 CloudStack – check content –

Apache CloudStack is open source software designed to deploy and manage large networks of virtual machines, as a highly available, highly scalable Infrastructure as a Service (IaaS) cloud computing platform. It uses existing hypervisors such as KVM, VMware vSphere, and XenServer/XCP for virtualization. In addition to its own API, CloudStack also supports the Amazon Web Services (AWS) API and the Open Cloud Computing Interface from the Open Grid Forum. [631]

CloudStack features like built-in high-availability for hosts and VMs, AJAX web GUI for management, AWS API compatibility, Hypervisor agnostic, snapshot management, usage metering, network management (VLAN's, security groups), virtual routers, firewalls, load balancers and multi-role support. [715]

70.19.14 CoreOS – check content –

[147] states that “CoreOS is a linux operating system used for clustered deployments.” CoreOS allows applications to run on containers. CoreOS can be run on clouds, virtual or physical servers. CoreOS allows the ability for automatic software updates in order to make sure containers in cluster are secure and reliable. It also makes managing large cluster environments easier. CoreOS provides open source tools like CoreOS Linux, etcd, rkt and flannel. CoreOS also has commercial products Kubernetes and CoreOS stack. In CoreOS linux service discovery is achieved by etcd, applications are run on Docker and process management is achieved by fleet.

70.19.15 rkt – check content –

rkt is a container manager developed by CoreOS [146] designed for Linux clusters. It is an alternative for Docker runtime and is designed for server environments with high security and composability requirement. It is the first implementation of the open container standard called *App Container* or *appc* specification but not the only one. It is a standalone tool that lives outside of the core operating system and can be used on variety of platforms such as Ubuntu, RHEL, CentOS, etc. rkt implements the facilities specified by the App Container as a command line tool. It allows execution of App Containers with pluggable isolation and also varying degrees of protection. Unlike Docker, rkt runs containers as un-privileged users making it impossible for attackers to break out of the containers and take control of the entire physical server. rkt's primary interface comprises a single executable allowing it easily integrate with existing init systems and also advanced cluster environments. rkt is open source and is written in the Go programming language [245].

70.19.16 VMware ESXi

VMware ESXi (formerly ESX) is an enterprise-class, type-1 hypervisor developed by VMware for deploying and serving virtual computers [699]. The name ESX originated as an abbreviation of Elastic Sky X. ESXi installs directly onto your physical server enabling it to be partitioned into multiple logical servers referred to as virtual machines. Management of VMware ESXi is done via APIs. This allows for an *agent-less* approach to hardware monitoring and system management. VMware also provides remote command lines, such as the vSphere Command Line Interface (vCLI) and PowerCLI, to provide command and scripting capabilities in a more controlled manner. These remote command line sets include a variety of commands for configuration, diagnostics and troubleshooting. For low-level diagnostics and the initial configuration, menu-driven and command line interfaces are available on the local console of the server [674].

70.19.17 vSphere and vCloud – check content –

vSphere was developed by VMware and is a cloud computing virtualization platform. [675] vSphere is not one piece of software but a suite of tools that contains software such as vCenter, ESXi, vSphere client and a number of other technologies. ESXi server is a type 1 hypervisor on a physical machine of which all virtual machines are installed. The vSphere client then allows administrators to connect to the ESXi and manage the virtual machines. The vCenter server is a virtual machine that is also installed on the ESXi server which is used in environments when multiple ESXi servers exist. Similarly, vCloud is also a suite of applications but for establishing an infrastructure for a private cloud. [103] The suite includes the vsphere suite, but also contains site recovery management for disaster recovery, site networking and security. Additionally, a management suite that can give a visual of the infrastructure to determine where potential issues might arise.

70.19.18 Amazon – check content –

FORMAT WRONG

Amazon's AWS (Amazon Web Services) is a provider of Infrastructure as a Service (IaaS) on cloud. It provides a broad set of infrastructure services, such as computing, data storage, networking and databases. One can leverage AWS services by creating an account with AWS and then creating a virtual server, called as an instance, on the AWS cloud. In this instance you can select the hard disk volume, number of CPUs and other hardware configuration based on your application needs. You can also select operating system and other software required to run your application. AWS lets you select from the countless services. Some of them are mentioned below:

- Amazon Elastic Computer Cloud (EC2) - Amazon Simple Storage Service (Amazon S3) - Amazon CloudFront - Amazon Relational Database Service (Amazon RDS) - Amazon SimpleDB - Amazon Simple Notification Service (Amazon SNS) - Amazon Simple Queue Service (Amazon SQS) - Amazon Virtual Private Cloud (Amazon VPC)

Amazon EC2 and Amazon S3 are the two core IaaS services, which are used by cloud application solution developers worldwide. [619]

Improve: all of them need bibentries

70.19.19 Azure

70.19.20 Google and other public Clouds

A public cloud is a scenario where a provider provides services such as infrastructure or applications to the public over the internet. Google cloud generally refers to services such as cloud print, connect, messaging, storage and platform [259]. Google cloud print allows a print-aware application on a device, installed on a network, to provide prints to any printer on that network. Cloud connect allows an automatic storage and synchronization of Microsoft word documents, power-points and excel sheets to Google docs while preserving the Microsoft office formats. In certain cases, developers require important notifications to be sent to applications targeting android operating system. Google cloud messaging provides such services. Google cloud platform allows the developers to deploy their mobile, web and backend solutions on a highly scalable and reliable infrastructure [260]. It gives developers a privilege of using any programming language. Google cloud platform provides a wide range of products and services including networking, storage, machine learning, big data, authentication and security, resource management, etc. In general, public clouds provide services to different end users with the usage of the same shared infrastructure [687]. Windows Azure services platform, Amazon elastic compute cloud and Sun cloud are few examples of public clouds.

70.19.21 Networking: Google Cloud DNS

Under the umbrella of google cloud platform, helps user to publish their domain using Google's infrastructure. It is highly scalable, low latency, high availability DNS service residing on infrastructure same as google.

It is build around projects a resource container, domain for access control, and billing configuration. Managed zones holds records for same DNS name. The resource record sets collection holds current state of the DNS that make up managed zones it is unmodifiable or cannot be modified easily and changes to record sets. It supports A address records, AAAA IPv6, CAA Certificate authority, CNAME canonical name, MX mail exchange, NAPTR naming authority pointer, NS Name server record, SOA start of authority, SPF Sender policy framework, SRV service locator, TXT text record.

-- check content --

70.19.22 Amazon Route 53

Amazon Route 53 is a DNS (Domain Name System) service that gives developers and businesses a reliable way to route end users to Internet applications. The number 53 refers to TCP or UDP port 53, where DNS server requests are addressed [25].

When using Route 53 as your DNS provider, in case of a recursion, the query of fetching an IP address (of a website or application) always goes to the closest server location to reduce query latency. The Route 53 server returns the IP address enabling the browser to load the website or application. Route 53 can also be used for registering domain names and arranging DNS health checks to monitor the server [688].

70.20 Cross-Cutting Functions

70.20.1 Monitoring

Ambari – check content –

Apache Ambari is an open source platform that enables easy management and maintenance of Hadoop clusters, regardless of cluster size. Ambari has a simplified Web UI and robust REST API for automating and controlling cluster operations. [31] illustrates Ambari to provide key benefits including easy installation, configuration, and management with features such as Smart Configs and cluster recommendations and Ambari Blueprints, to provide repeatable and automated cluster creation. Ambari provides a centralized security setup that automates security capabilities of clusters. Ambari provides a holistic view for cluster monitoring and provides visualizations for operation metrics. [29] provides documentation about Ambari, including a quick start guide for installing a cluster with Ambari. [30] provides the project documents for ambari on github.

Ganglia

Ganglia is a scalable distributed monitoring system for high-performance computing systems (clusters and grids). It is a BSD-licensed open-source project that grew out of the University of California, Berkeley Millennium Project which was initially funded in large part by the National Partnership for Advanced Computational Infrastructure (NPACI) and National Science Foundation RI Award EIA-9802069 [232].

It relies on a multicast-based listen/announce protocol to monitor state within clusters. It uses a tree of point-to-point connections amongst representative cluster nodes to unite clusters and aggregate their state [233]. It leverages technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on thousands of clusters around the world, handling clusters with 2000 nodes.

Nagios (www-nagios -- check content --)

Nagios is a platform, which provides a set of software for network infrastructure monitoring. It also offers administrative tools to diagnose when failure events happen, and to notify operators when hardware issues are detected. Specifically, illustrates that Nagios is consist of modules including [353]: a core and its dedicated tool for core configuration, extensible plugins and its frontend. Nagios core is designed with scalability in mind. Nagios contains a specification language allowing for building an extensible monitoring systems. Through the Nagios API components can integrate with the Nagios core services. Plugins can be developed via static languages like C or script languages. This mechanism empowers Nagios to monitor a large set of various scenarios yet being very flexible. [337] Besides its open source components, Nagios also has commercial products to serve needing clients.

Inca – check content –

Inca is a grid monitoring [681] software suite. It provides grid monitoring features. These monitoring features provide operators failure trends, debugging support, email notifications, environmental issues etc. [321]. It enables users to automate the tests which can be executed on a periodic basis. Tests can be added and configured as and when needed. It helps users with different portfolios like system administrators, grid operators, end users etc Inca provides user-level grid monitoring. For each user it stores results as well as allows users to deploy new tests as well as share the results with other users. The incat web ui allows users to view the status of test, manage test and results. The

architectural blocks of inca include report repository, agent, data consumers and depot. Reporter is an executable program which is used to collect the data from grid source. Reporters can be written in perl and python. Inca repository is a collection of pre build reporters. These can be accessed using a web url. Inca repository has 150+ reporters available. Reporters are versioned and allow automatic updates. Inca agent does the configuration management. Agent can be managed using the incat web ui. Inca depot provides storage and archival of reports. Depot uses relational database for this purpose. The database is accessed using hibernate backend. Inca web UI or incat provides real time as well as historical view of inca data. All communication between inca components is secured using SSL certificates. It requires user credentials for any access to the system. Credentials are created at the time of the setup and installation. Inca's performance has been phenomenal in production deployments. Some of the deployments are running for more than a decade and has been very stable. Overall Inca provides a solid monitoring system which not only monitors but also detects problems very early on.

70.20.2 Security and Privacy

InCommon – check content –

The mission of InCommon is to “create and support a common trust framework for U.S. education and research. This includes trustworthy shared management of access to on-line resources in support of education and research in the United States.” [322] This mission ultimately is a simplification and an elimination of the need for multiple accounts across various websites that are at risk of data spills or misuse. In the academic setting, this helps assist researchers to focus on their area of study, and enabling the cross collaboration which is happening on a global scale. Currently any two and four year higher education institution that is accredited is eligible for joining InCommon.

Eduroam

Eduroam is an initiative started in the year 2003 when the number of personal computers with in the academia are growing rapidly. The goal is to solve the problem of secure access to WI-FI due to increasing number of students and research teams becoming mobile which was increasing the administrative problems for provide access to WI-FI. Eduroam provides any user from an eduroam participating site to get network access at any institution connected through eduroam. According to the organization it uses a combination of radius-based infrastructure with 802.1X standard technology to provide roaming access across research and educational networks. The role of the RADIUS hierarchy is to forward user credentials to the users home institution where they can be verified. This proved to be a successful solution when compared to other traditional ways like using MAC-address, SSID, WEP, 802.1x (EAP-TLS, EAP-TTLS), VPN Clients, Mobile-IP etc which have their own shortcomings when used for this purpose [211]. Today by enabling eduroam users get access to internet across 70 countries and tens of thousands of access points worldwide [181].

OpenStack Keystone – check content –

[368] Keystone is the identity service used by OpenStack for authentication (authN) and high-level authorization (authZ). There are two authentication mechanisms in Keystone, UUID token, and PKI. Universally unique identifier (UUID) is a 128-bit number used to identify information (user). Each application after each request of the client checks token validity online. PKI was introduced later and improved the security of Keystone [155]. In PKI, each token has its own digital signature that can be checked by any service and OpenStack application with no necessity to ask for Keystone database [134].

Thus, Keystone enables ensuring user's identity with no need to transmit its password to applications. It has recently been rearchitected to allow for expansion to support proxying external services and AuthN/AuthZ mechanisms such as OAuth, SAML and openID in future versions [473].

LDAP

LDAP stands for Lightweight Directory Access Protocol. It is a software protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the Internet or on corporate internet. [620]

LDAP is a lightweight (smaller amount of code) version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network. In a network, a directory tells you where in the network something is located. On TCP/IP networks (including the Internet), the domain name system (DNS) is the directory system used to relate the domain name to a specific network address (a unique location on the network). However, you may not know the domain name. LDAP allows you to search for an individual without knowing where they're located (although additional information will help with the search). An LDAP directory can be distributed among many servers. Each server can have a replicated version of the total directory that is synchronized periodically. An LDAP server is called a Directory System Agent (DSA). An LDAP server that receives a request from a user takes responsibility for the request, passing it to other DSAs as necessary, but ensuring a single coordinated response for the user.

Sentry – check content –

“Apache Sentry [69] is a granular, role-based authorization module for Hadoop. Sentry provides the ability to control and enforce precise levels of privileges on data for authenticated users and applications on a Hadoop cluster. Sentry currently works out of the box with Apache Hive, Hive Metastore/HCatalog, Apache Solr, Impala and HDFS (limited to Hive table data). Sentry is designed to be a pluggable authorization engine for Hadoop components. It allows the client to define authorization rules to validate a user or application's access requests for Hadoop resources. Sentry is highly modular and can support authorization for a wide variety of data models in Hadoop.”

Sqrl

OpenID – check content –

OpenID is an authentication protocol that allows users to log in to different websites, which are not related, using the same login credentials for each, i.e. without having to create separate id and password for all the websites. The login credentials used are of the existing account. The password is known only to the identity provider and nobody else which relieves the users' concern about identity being known to an insecure website. [467] It provides a mechanism that makes the users control the information that can be shared among multiple websites. OpenID is being adopted all over the web. Most of the leading organizations including Microsoft, Facebook, Google, etc. are accepting the OpenIDs [468]. It is an open source and not owned by anyone. Anyone can use OpenID or be an OpenID provider and there is no need for an individual to be approved.

SAML OAuth

As explained in [556], Security Assertion Markup Language (SAML) is a secured XML based communication mechanism for communicating identities between organizations. The primary use case of SAML is Internet SSO. It eliminates the need to maintain multiple authentication credentials

in multiple locations. This enhances security by eliminating opportunities for identity theft/Phishing. It increases application access by eliminating barriers to usage. It reduces administration time and cost by excluding the effort to maintain duplicate credentials and helpdesk calls to reset forgotten passwords. Three entities of SAML are the users, Identity Provider (IdP-Organization that maintains a directory of users and an authentication mechanism) and Service Provider (SP-Hosts the application /service). User tries to access the application by clicking on a link or through an URL on the internet. The Federated identity software running in the IdP validates the user's identity and the user is then authenticated. A specifically formatted message is then communicated to the federated identity software running at SP. SP creates a session for the user in the target application and allows the user to get direct access once it receives the authorization message from a known identity provider.

70.20.3 Distributed Coordination

Google Chubby – check content –

Chubby Distributed lock service [258] is intended for use within a loosely-coupled distributed system consisting of moderately large numbers of small machines connected by a high-speed network. Asynchronous consensus is solved by the Paxos protocol. The implementation in Chubby is based on coarse grained lock server and a library that the client applications link against. As per the 2016 paper [16], an open-source implementation of the Google Chubby lock service was provided by the Apache ZooKeeper project. ZooKeeper used a Paxos-variant protocol Zab for solving the distributed consensus problem. Google stack and Facebook stack both use versions of zookeeper.

Zookeeper – check content –

Zookeeper provides coordination services to distributed applications. It includes synchronization, configuration management and naming services among others. The interfaces are available in Java and C [774]. The services themselves can be distributed across multiple Zookeeper servers to avoid single point of failure. If the leader fails to answer, the clients can fall-back to other nodes. The state of the cluster is maintained in an in-memory image along with a persistent storage file called znode by each server. The cluster namespace is maintained in a hierarchical order. The changes to the data are totally ordered [775] by stamping each update with a number. Clients can also set a watch on a znode to be notified of any change [776]. The performance of the ZooKeeper is optimum for read-dominant workloads. It's maintained by Apache and is open-source.

Giraffe

Giraffe is a scalable distributed coordination service. Distributed coordination is a media access technique used in distributed systems to perform functions like providing group membership, gaining lock over resources, publishing, subscribing, granting ownership and synchronization together among multiple servers without issues. Giraffe was proposed as alternative to coordinating services like Zookeeper and Chubby which were efficient only in read-intensive scenario and small ensembles. To overcome this three important aspects were included in the design of Giraffe [571]. First feature is Giraffe uses interior-node joint trees to organize coordination servers for better scalability. Second, Giraffe uses Paxos protocol for better consistency and to provide more fault-tolerance. Finally, Giraffe also facilitates hierarchical data organization and in-memory storage for high throughput and low latency.

JGroups

70.21 Message and Data Protocols

70.21.1 MQTT

70.21.2 Avro

Apache Avro is a data serialization system, which provides rich data structures, remote procedure call (RPC), a container file to store persistent data and simple integration with dynamic languages [41]. Avro depends on schemas, which are defined with JSON. This facilitates implementation in other languages that have the JSON libraries. The key advantages of Avro are schema evolution - Avro will handle the missing/extra/modified fields, dynamic typing - serialization and deserialization without code generation, untagged data - data encoding and faster data processing by allowing data to be written without overhead.

70.21.3 Thrift – check content –

The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages. [580] It includes a complete stack for creating clients and servers. It includes a server infrastructure to tie the protocols and transports together. There are blocking, non-blocking, single and multithreaded servers available. Thrift was originally developed at Facebook, it was open sourced in April 2007 and entered the Apache Incubator in May, 2008. It became an Apache TLP in October, 2010. [645]

70.21.4 Protobuf – check content –

Protocol Buffer [515] is a way to serialize structured data into binary form (stream of bytes) in order to transfer it over wires or for storage. It is used for inter application communication or for remote procedure call (RPC). It involves a interface description that describes the structure of some data and a program that can generate source code or parse it back to the binary form. It emphasizes on simplicity and performance over xml. Though xml is more readable but requires more resources in parsing and storing. This is developed by Google and available under open source licensing. The parser program is available in many languages including java and python.

-- check content --

70.22 Technologies To Be Integrated

70.22.1 Snort – check content –

[737] Snort is a Network Intrusion Prevention System (NIPS) and Network Intrusion Detection System (NIDS). Snort's open source network-based intrusion detection system (NIDS) has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort performs protocol analysis, content searching and matching. These basic services have many purposes including application-aware triggered quality of service, to de-prioritize bulk traffic when latency-sensitive applications are in use. The program can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans. Snort can be configured

in three main modes: sniffer, packet logger, and network intrusion detection. In sniffer mode, the program will read network packets and display them on the console. In packet logger mode, the program will log packets to the disk. In intrusion detection mode, the program will monitor network traffic and analyze it against a rule set defined by the user. The program will then perform a specific action based on what has been identified.

70.22.2 Fiddler

Fiddler is an HTTP debugging proxy server application. Fiddler captures HTTP and HTTPS traffic and logs it for the user to review by implementing man-in-the-middle interception using self-signed certificates. Fiddler can also be used to modify (fiddle with) HTTP traffic for troubleshooting purposes as it is being sent or received.[5] By default, traffic from Microsoft's WinINET HTTP (S) stack is automatically directed to the proxy at runtime, but any browser or Web application (and most mobile devices) can be configured to route its traffic through Fiddler [726].

70.22.3 Zeppelin – check content –

Apache Zeppelin [770] provides an interactive environment for big data data analytics on applications using distributed data processing systems like Hadoop and Spark. It supports various tasks like data ingestion, data discovery, data visualization, data analytics and collaboration. Apache Zeppelin provides built-in Apache Spark integration and is compatible with many languages/data-processing backends like Python, R, SQL, Cassandra and JDBC. It also supports adding new language backend. Zeppelin also lets users to collaborate by sharing their Notebooks, Paragraph and has option to broadcast any changes in realtime.

70.22.4 Open MPI – check content –

The Open MPI Project [637] is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers. Open MPI [226] provides functionality that has not previously been available in any single, production-quality MPI implementation, including support for all of MPI-2, multiple concurrent user threads, and multiple options for handling process and network failures.

70.22.5 Apache Tomcat – check content –

Apache tomcat is an open source java servlet container. [479] It is used in IT industry as a HTTP web server which listens to the requests made by web client and send responses. The main components of tomcat are cataline, coyote and jasper. The most stable version of Apache Tomcat server is version 8.5.11. Apache tomcat is released under Apache License version 2. [706] As it is cross platform, it can run in any platform or OS like Windows, UNIX, AIX or SOLARIS etc. It is basically an integral part of many java based web application.

70.22.6 Apache Beam – check content –

Apache Beam attempts to abstract away the need to write code for multiple data-oriented workflows, e.g., batch, interactive and streaming, as well as multiple big data tools, e.g., Storm, Spark and

Flink. Instead, Beam attempts to automagically map a dataflow process written in Java or Python to the target runtime environment via *runners*. As a result, switching a data processing routine from Spark to Flink only requires changing the target runtime environment as opposed to re-writing the entire process [504] (perhaps in a completely different language). Google contributed its Dataflow SDK, the Dataflow model and three runners [745] to the Apache Software Foundation in the first half of 2016. The ASF elevated Beam to a Top-Level project in January 2017. Jean-Baptiste Onofre of French tech company Talend, and a frequent Apache project contributor, champions the project. [460] It should be grouped with the technologies in the *Interoperability* section.

70.22.7 Cloudability – check content –

Cloudability is a financial management tool for analyzing and monitoring all cloud expenses across an organization. It can be used for cost monitoring, usage rightsizing, reserved instance planning, cost allocation, role-based visibility. It aggregates expenditures into reports, helps identify opportunities for reducing costs, offers budget alerts and recommendations via SMS and email, and provides APIs for connecting cloud billing and usage data to any business or financial system. [131]

70.22.8 CUDA – check content –

It is a parallel computing platform and application programming interface (API) model created by Nvidia. It allows software developers to use a CUDA-enabled graphics processing unit for general purpose processing. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels. CUDA platform has advantages such as scattered reads i.e the code can read from arbitrary addresses in memory, unified virtual memory, unified memory, faster downloads and readbacks to and from the GPU and full support for integer and bitwise operations. [722]. CUDA is used for accelerated rendering of 3D graphics, accelerated interconversion of video file formats, encryption, decryption and compression of files. It is also used for distributed calculations, face recognition and distributed computing. [722]

70.22.9 Blaze – check content –

Blaze library translates NumPy/Pandas-like syntax to data computing systems (e.g. database, in-memory, distributed-computing). This provides Python users with a familiar interface to query data in a variety of other data storage systems. One Blaze query can work across data ranging from a CSV file to a distributed database.

Blaze presents a pleasant and familiar interface regardless of what computational solution or database we use (e.g. Spark, Impala, SQL databases, No-SQL data-stores, raw-files). It mediates the users interaction with files, data structures, and databases, optimizing and translating the query as appropriate to provide a smooth and interactive session. It allows the data scientists and analyst to write their queries in a unified way that does not have to change because the data is stored in another format or a different data-store. [142]

70.22.10 CDAP – check content –

CDAP [747] stands for Cask Data Application Platform. CDAP is an application development platform using which developers can build, deploy and monitor applications on Apache Hadoop. In a typical CDAP application, a developer can ingest data, store and manage datasets on Hadoop, perform batch mode data analysis, and develop web services to expose the data. They can also

schedule and monitor the execution of the application. This way, CDAP enables the developers to use single platform to develop the end to end application on Apache Hadoop.

CDAP documentation [748] explains the important CDAP concepts of CDAP Dataset, CDAP Application and CDAP Services. CDAP Datasets provide logical abstraction over the data stored in Hadoop. CDAP Applications provide containers to implement application business logic in open source processing frameworks like map reduce, Spark and real time flow. CDAP applications also provide standardize way to deploy and manage the apps. CDAP Services provide services for application management, metadata management, and streams management. CDAP can be deployed on various Hadoop Platforms such as Apache Hadoop, Cloudera Hadoop, Hortonworks Hadoop and Amazon EMR. CDAP sample apps [244] provide explain how to implement apps on CDAP platform.

70.22.11 Apache Arrow – check content –

Apache arrow allows execution engines to utilize what is known as Single Input multiple data (SIMD). [36] This SIMD is an operation that allows modern processors to take advantage of this engine. Performance is enhanced by grouping relevant data as close as possible in a column format. Many programming languages are supported such a Java, C, C++, Python and it is anticipated that languages will be added as it grows. It is still in early development but has released a 0.1.0 build.

70.22.12 OpenRefine

OpenRefine (formerly GoogleRefine) is an open source tool that is dedicated to cleaning messy data. With the help of this user-friendly tool you can explore huge data sets easily and quickly even if the data is a little unstructured. It allows you to load data, understand it, clean it up, reconcile it, and augment it with data coming from the web [470]. It operates on rows of data which have cells under columns, which is very similar to relational database tables. One OpenRefine project is one table. The user can filter the rows to display using facets that define filtering criteria. most operations in OpenRefine are done on all visible rows: transformation of all cells in all rows under one column, creation of a new column based on existing column data, etc. All actions that were done on a dataset are stored in a project and can be replayed on another dataset. It has a huge community with lots of contributors meaning that the software is constantly getting better and better.

70.22.13 Apache OODT – check content –

Apache Object Oriented Data Technology (OODT) [461] is a distributed data management technology that helps to integrate and archive your processes, your data, and its metadata. OODT allows to generate, process, manage and analyze distributed and heterogeneous data enabling integration of different, distributed software systems. Apache OODT uses structured XML-based capturing of the processing pipeline which is used to create, edit, manage and provision workflow and task execution. OODT is written in Java programming language and provides its own set of APIs for storing and processing data. [462] It provides three core services. A File Manager is responsible for tracking file locations, their metadata, and for transferring files from a staging area to controlled access storage. A Workflow Manager captures control flow and data flow for complex processes, and allows for reproducibility and the construction of scientific pipelines. A Resource Manager handles allocation of workflow tasks and other jobs to underlying resources, e.g., Python jobs go to nodes with Python installed on them similarly jobs that require a large disk or CPU are properly sent to those nodes that fulfill those requirements. OODT is now supported with Apache Mesos

and Grid Computing which can allow for creating of highly distributed, scalable data platforms that can process large amounts of data. OODT technology is used in NASA's Jet Propulsion Laboratory.

70.22.14 Omid – check content –

Omid is a “flexible, reliable, high performant and scalable ACID transactional framework” [59] for NoSQL databases, developed by Yahoo for HBase and contributed to the Apache community. Most NoSQL databases, do not natively support ACID transactions. Omid employs a lock free approach from concurrency and can scale beyond 100,000 transactions per second. At Yahoo, millions of transactions per day are processed by Omid. [492].

Omid is currently in the Apache Incubator. All projects accepted by the Apache Software Foundation (ASF) undergo an incubation period until a review indicates that the project meets the standards of other ASF projects [57]

70.22.15 Apache Ant

Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks. Ant is written in Java. Users of Ant can develop their own “antlibs” containing Ant tasks and types, and are offered a large number of ready-made commercial or open-source “antlibs”. Ant is extremely flexible and does not impose coding conventions or directory layouts to the Java projects which adopt it as a build tool. Software development projects looking for a solution combining build tool and dependency management can use Ant in combination with Apache Ivy. The Apache Ant project is part of the Apache Software Foundation [625].

70.22.16 LXD – check content –

LXD is a demon processes established to manage the containers. It can be understood as hypervisor for linux containers. It is implemented by exporting RESTful API for libxl to the remote network or local unix socket. [667]. It implements the under privilized containers by default adding more security. It works with Image based work flow supports online snapshooting and live container migration. [401]. It was build with aim of providing VM like virtualization with container like performance. [636]

70.22.17 Wink – check content –

Apache wink [66] provides a framework to develop and use RESTful web services. It implements using JAX-RS v1.1 specification. The project provides server module which integrates with all popular web servers and a client module which can used to write RESTful web services. This project will be integrated with Geronimo and other opensource REST projects to build a vendor neutral community. Currently IBM and HP have taken lead. IBM is writing a full JAX-RS implementation while HP is working on RESTful SDK for client and server components. Portion of initial project was also taken from Apache CXF which uses other Apache components like commons-codec, commons-logging, Apache-Abdera. Apache wink will simply web services development using one single standard.

70.22.18 Apache Apex

Apache Apex is “a YARN (Hadoop 2.0)-native platform that unifies cloud and batch processing” [700]. This project was developed under Apache License 2.0 and was driven by Data Torrent. It can be used for processing both streams of data and static files making it more relevant in the context of present day internet and social media. It is aimed at leveraging the present Hadoop platform and reducing the learning curve for development of applications over it. It is aimed at It can used through a simple API. It enables reuse of code by not having to make drastic changes to the applications by providing interoperability with existing technology stack. It leverages the existing Hadoop platform investments.

Apart from the Apex core component, it also has Apex Malhar which provides a library of connectors and logic functions. It provides connectors to existing file systems, message systems and relational, NoSQL and Hadoop databases, social media. It also provides a library of compute operators like Machine Learning, Stats and Math, Pattern Marching, Query and Scripting, Stream manipulators, Parsers and UI & Charting operators [363].

70.22.19 Apache Knox – check content –

According to [50], “the Apache Knox Gateway is a REST API Gateway for interacting with Apache Hadoop clusters.” REST stands for Representational State Transfer and is web architectural style designed for distributed hypermedia systems and defines a set of constraints. [204] API Gateways manage concerns related to “Authentication, Transport Security, Load-balancing, Request Dispatching (including fault tolerance and service discovery), Dependency Resolution, Transport Transformations.” [493] Although every Apache Hadoop cluster has its own set of REST APIs, Knox will represent all of them as “a single cluster specific application context path.” [50] Knox protects Apache Hadoop clusters, by way of its gateway function, by aiding “the control, integration, monitoring and automation of critical administrative and analytical needs.” [50] Some Apache Hadoop Services that integrate with Knox are, “Ambari, WebHDFS (HDFS), Templeton (Hcatalog), Stargate (Hbase), Oozie, Hive/JDBC, Yarn RM, [and] Storm.” [50] Apache Knox has a configuration driven method to aid in the addition of new routing services. [50] This allows support for new and custom Apache Hadoop REST APIs to be added to the Knox gateway quickly and easily. [50] This technology would be best placed under the interoperability category.

70.22.20 Apache Apex

The Apex platform is designed to process real-time events with streaming data natively in Hadoop. The platform handles application execution, dynamic scaling, state checkpointing and recovery, etc. This allows the users to focus on writing their application logic without mixing operational and functional concerns [39]. In the platform, building a streaming application is easy and intuitive.

An application may consist of one or more operators each of which define some logical operation to be done on the tuples arriving at the operator. These operators are connected together to form streams. A streaming application is represented by a DAG that consists of operators and streams [40]. The Apex platform comes with support for web services and metrics. This enables ease of use and easy integration with current data pipeline components. DevOps teams can monitor data in action using existing systems and dashboards with minimal changes, thereby easily integrating with the current setup. With different connectors and the ease of adding more connectors, Apex easily integrates with an existing dataflow [121].

70.22.21 Robot Operating System (ROS)

The aptly-named *Robot Operating System*, or ROS, provides a framework for writing operating systems for robots. ROS offers “a collection of tools, libraries, and conventions [meant to] simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms” [466]. ROS’ designers, the Open Source Robotics Foundation, hereinafter OSRF or the Foundation, attempt to meet the aforementioned objective by implementing ROS as a modular system. That is, ROS offers a core set of features, such as inter-process communication, that work with or without pre-existing, self-contained components for other tasks.

The OSRF designed ROS as a distributed, modular system. The OSRF maintains a subset of essential features for ROS, i.e., *ROS core*, to provide an extensible platform for other roboticists. The Foundation also coordinates the maintenance and distribution of a vast array of ROS add-ons, referred to as modules. ROS’ core consists of the following components: (a) communications infrastructure; (b) robot-specific features; and, (c) tools. The modules, analogous to packages in Linux repositories or libraries in other software packages such as *R*, provide solutions for numerous robot-related problems. General categories include (a) drivers, such as sensor and actuator interfaces; (b) platforms, for steering and image processing, etc.; (c) algorithms, for task planning and obstacle avoidance; and, (d) user interfaces, such as tele-operation and sensor data display [439].

70.22.22 Apache Flex – check content –

Apache Flex [209] is an open source application framework for building and maintaining mobile and web applications that deploy consistently on multiple browsers, desktops and mobile devices. It was initially developed by Macromedia and then acquired by Adobe Systems. It was later donated to the Apache Software Foundation in 2011 [210]. It can pull data from multiple back-end sources such as Java, Spring, PHP, Ruby, .NET, Adobe ColdFusion, and SAP and display it visually allowing users to drill down into the data for deeper insight and even change the data and have it automatically updated on the back end [696].

70.22.23 Apache Ranger – check content –

Apache Ranger [746] is open source software project designed to provide centralized security services to various components of Apache Hadoop. Apache Hadoop provides various mechanism to store, process and access the data. Each Apache tool has its own security mechanism. This increases administrative overhead and is also error prone. Apache Ranger fills this gap to provide a central security and auditing mechanism for various Hadoop components [759]. Using Ranger, Hadoop administrators can perform security administration tasks using a central UI or Restful web services. He can define policies which enable users/user-groups to perform specific action using Hadoop components and tools. Ranger provides role based access control for datasets on Hadoop at column and row level. The blog article [760] explains that the row level filtering and dynamic data masking are most important features of Apache Ranger. Ranger also provides centralized auditing of user acces and security related administrative actions.

70.22.24 Google Cloud Machine Learning

Google Cloud Machine Learning is a Google’s cloud based managed system for building machine learning model, capable to work on any type and volume of data. User can create their own machine learning model using GoogleTensorFlow framework, which helps to use the range of

Google products from Google Photos to Google Cloud Speech. We can build our machine learning model regardless the size, google will managed it infrastructure according to requirement. User can immediately host the created model and start predicting on new data [264]. Cloud Machine Learning provides two important things:

- Help user to train the machine learning model at large scale with the help of TensorFlow training application.
- User can host the trained model on cloud, this will help to use the large and new data available on cloud, which help in creating good model.

Google CloudML will help user to focus on model instead of hardware configuration and resource management [265].

70.22.25 Karajan

Karajan is used to allow users to describe various workflows using XML [507]. It also uses a custom yet user friendly language called K. The advantages of using XML and K is that we can use Directed Acyclic Graphs (DAGs) to describe hierarchical workflows. Besides, it is also very easy to handle concurrency using trivial programming constructs like if/while orders. It can also use tools such as Globus GRAM for parallel or distributed execution of various workflows. From an architectural perspective, Karajan mainly consists of three components: Workflow engine, that monitors the execution and is responsible for the higher level interaction with higher level components like the Graphical User Interface Module (GUI) for the description of various workflows; Workflow service, that is used to allow the execution of various workflows using specific functionalities that can be accessed by the workflow engine using specific libraries; and the Checkpointing subsystem that monitors and checks the current state of the workflow. Karajan is typically used as a scientific workflow scheduling technique for various Big Data platforms.

The Karajan code, that can be obtained from Java CoG Kit CVS archive has two interfaces: the command line interface (CLI) and the GUI. The CLI can be accessed via bin/karajan and provides a minimalist interface that is non-interactive and doesn't provide much feedback on the execution status. As against this, the GUI can be accessed via bin/karajan-gui and provides an enriched interface that provides visual features to determine the execution status besides being interactive in real time [386].

70.23 Exercise

TechList.1: In class you will be given an HID and you will be assigned a number of technologies that you need to research and create a summary as well as one or more relevant references to be added to the Web page. All technologies for TechList.1 are marked with a (1) behind the technology. An example text is given for Nagios in this page. Please create a pull request with your responses. You are responsible for making sure the request shows up and each commit is using gitchangelog in the commit message::

```
new:usr: added paragraph about <PUTTECHHERE>
```

You can create one or more pull requests for the technology and the references. We have created in the referens file a placeholder using your HID to simplify the management of the references while avoiding conflicts. For the technologies you are responsible to investigate them and write an academic summary of the technology. Make sure to add your reference to refs.bib. Many technologies may have additional references than the Web page. Please add the most important once while limiting it to three if you can. Avoid plagiarism and use

proper quotations or better rewrite the text.

You must look at ?? to successfully complete the homework

A video about this homework is posted at <https://www.youtube.com/watch?v=roi7vezNmfo> showing how to do references in emacs and jabref, it shows you how to configure git, it shows you how to do the fork request while asking you to add ‘new:usr’ to the commit messages. As this is a homework related video we put a lot of information in it that is not only useful for beginners. We recommend you watch it.

This homework can be done in steps. First you can collect all the content in an editor. Second you can create a fork. Third you can add the new content to the fork. Fourth you can commit. Fifth you can push. Six if the TAs have commend improve. The commit message must have new:usr: at the beginning.

While the Nagios entry is a good example (make sure grammer is ok the Google app engine is an example for a bad entry.

Do Techlist 1.a 1.b 1.c first. We will assign Techlist 1.d and TechList 2 in February.

TechList.1.a: Complete the pull request with the technologies assigned to you. Details for the assignment are posted in Piazza. Search for TechList.

TechList.1.b: Identify how to cite. We are using *scientific* citation formats such as IEEEtran, and ACM. We are **not** using citation formats such as Chicago, MLA, or ALP. The later are all for non scientific publications and thus of no use to us. Also when writing about a technology do not use the names of the person, simply say something like. In [1] the definition of a turing machine is given as follows, ... and do not use elaborate sentences such as: In his groundbreaking work conducted in England, Allan Turing, introduced the turing machine in the years 1936-37 [2]. Its definition is base on ... The difference is clear, while the first focusses on results and technological concepts, the second introduces a colorful description that is more suitable for a magazine or a computer history paper.

TechList 1.c: Learn about plagiarism and how to avoid it. Many Web pages will conduct self advertisement while adding suspicious and subjective adjectives or phrases such as cheaper, superior, best, most important, with no equal, and others that you may not want to copy into your descriptions. Please focus on facts, not on what the author of the Web page claims.

TechList 1.d: Identify technologies from the Apache Project <https://projects.apache.org/> or other Big Data related Web pages and projects that are not yet listed here. Add them at the end of the Technologies page under the :ref:‘New Technologies <new-techs>‘ section, together with a description and appropriate references just like you did for your list of technologies in TechList 1a-1c. As part of your paragraph, please suggest a section where you think is best to add the technologies. Once the new technologies have been submitted, the AIs will integrate them in the appropriate sections. Please, only add new techs to the last section, otherwise it will be easy to introduce conflicts in the file.

TechList.2: In this hopweork we provide you with additional technologies that you need to complete. They are marked with (2) in the :doc:‘HID Assignment page <hids-techs>‘.

TechList.3: Identify technologies that are not listed here and add them. Provide a description and a reference just as you did before. Before you add a technology, verify that it is not on the **new technologies** list already. Duplicated entries will be merged.

TechList.4: For useful information on how to correctly create BibTeX entries, see and contribute to :ref:‘these open discussion threads Piazza <bibtex-discussions>‘.



74	ESP8266	771
75	Raspberry PI 3	783
75.1	Raspberry PI for IOT (Gregor)	
75.2	Hardware	
75.3	Installation	
75.4	Configure	
75.5	Update	
75.6	change python version	
75.7	install 3.6.1	
75.8	install cloudmesh.pi	
75.9	Sensors (Jon)	
75.10	Notes To integrates	
75.11	VLC on OSX	
75.12	Camera on Pi	
75.13	Streaming video	
75.14	Linux Commandline	
75.15	Enable SPI	
75.16	RTIMULib2	
75.17	Compile RTIMULib Apps	
75.18	Camera	
75.19	Lessons and Projects	
75.20	OTHER TO BE INTEGRATED	
75.21	Web Server	
76	Dexter	793
76.1	Creating an SD Card	
77	GrovePi Modules	795
77.1	Introduction	
77.2	LED	
77.3	Buzzer	
77.4	Relay	
77.5	Light Sensor	
77.6	Rotary Angle Sensor	
77.7	Barometer	
77.8	Distance Sensor	
77.9	Temperature Sensor	
77.10	Heartbeat Sensor	
77.11	Joystick	
77.12	LCD Screen	
77.13	Moisture Sensor	
77.14	Water Sensor	
77.15	Sensors	
78	VNC	803
78.1	Setting up VNC	
79	Turtle Graphics	805
79.1	Demo	
79.2	Program example	
79.3	Shape	
79.4	Links	
79.5	Robot Dance Simulator	
79.6	Scratch	
79.7	MBlock	
80	Tools	809
80.1	Markdown	
80.2	Aquamacs	
80.3	Bash	
80.4	Arduino	
80.5	OSX Terminal	



71. Introduction

Internet of Things is one of the driving forces in the modernisation of today's world. It is based on connecting *things* to the internet to create a more aware world that can be interfaced with. This not only includes us humans, but any *thing* that can interact with other things. It is clear that such a vision of interconnected devices will result in billions of devices to communicate with each other. Some of them may only communicate small number of items, while others will communicate a large amount. Analysis of this data is dependent on the capability of the *thing*. If it is too small the analysis can be conducted on a remote server or cloud while information to act are fed back from the device. In other cases the device may be completely autonomous and does not require any interaction. Yet in other cases the collaborative information gathered from such devices is used to derive decisions and actions.

Within this section we are trying to provide you with a small glimpse into how IoT devices function and can be utilized on small projects. Ideally if the class has all such a device we could even attempt to build a cloud based service that collects and redistributes the data.

To keep things simple we are not providing a general introduction in IoT. For that we offer other classes. However, we will introduce you to two different devices. These are

- esp8266
- Raspberry Pi

The reasons we chose them is that

1. they are cheap,
2. we can program both in python allowing us to use a single programming language for all projects and assignments, and
3. they are sufficiently powerful and we can conduct real projects with them beyond toy projects.
4. the devices, especially the Raspberry PI can be used to also learn Linux in case you do not have access to a linux computer. Please note however the raspberry will have memory and

space limitations that you need to deal with.

Projects that you can do to test the devices are

esp8266 (easy-moderate, small memory):

1. a LED blinker
2. a dendrite
3. a robot fish
4. a fish swarm
5. a robot swarm
6. an activity of your desire

Raspberry Pi (easy-moderate, 32GB space limitation):

1. a LED blinker
2. a robot car
3. a robot car with camera
4. a temperature service
5. a docker cluster

Crazyflie 2.0 (difficult):

1. programming a drone
2. programming a drone swarm

Please note that for those at IU we do have a Lab in which you can use some of the devices pointed out here. You can arrange for accessing the infrastructure or you simply can buy it for yourself.

We have a hardware page that summarizes what you need. In case you want to work on a swarm, we do have positioning sensors that simplify that task.

In general we think that these platforms provide a wonderful introduction into IoT and where it will move to. Such platforms were just a decade ago not powerful enough or too expensive. However today they provide a serious platform for developers. Sensors are available easily as most Android comparable sensors can be used.

Before we jump right into programming the devices, we like to point out that we did not choose to use Arduinos, as their price advantage is no longer valid. We also find that esp8266 and Raspberry can interface with most sensors. Having the ability to easily use WiFi however is our primary reason for using them. Furthermore being able to attach a camera to the Raspberry is just superb. Image analysis will be one of the near term future drivers for big data.



72. Hardware for IoT Projects

When teaching programming you may find yourself in a situation that things can be done on your computer, but you may not want to install programs that help you to learn programming on your computer. However, we have a solution (or several) for you. We will have some fun with hardware for IoT that at the same time can be used to teach you some very elementary skills in programming. However, if you would rather use your computer you certainly can do this too.

We see the following arguments for using IoT hardware:

- You will have fun with inexpensive hardware
- You will get hands on experience with IOT devices
- You will learn how to program in python
- You can keep your current computer unchanged
- You will get experience with two platforms esp8266 and Raspberry PI 3
- You can customise your choices by conducting some fun projects.
- You have the opportunity to find alternative hardware choices such as the WiPy or the ESP32. You may find cheaper or better alternatives if you buy kits when they are available. And learn in getting an overview about such devices and kits.

Note: Ordering from overseas suppliers may take significant time, so make sure to plan ahead. Prices given here are done to provide an estimate, they may vary.

72.1 Raspberry Pi 3

The raspberry PI 3 is a very good development platform. With its base price of \$35 it is quite a bargain. You will need some additional components to make sure you can use it. Please be reminded to never connect or power the raspberry with your computers USB port. It draws some significant amperage and we do not want you to destroy your computer. We recommend that you buy a certified power adapter. The price is so cheap that you could even create your own

mini cluster as a project. We do not recommend any older versions of Raspberry as they are less powerful and do not contain built-in Bluetooth or WiFi.

Configuration:

- \$37.50 [Pi 3](#)
- \$7.69 [Case](#)
- \$7.99 [Power Adapter](#) This is an aftermarket poweradapter. Lots uof us use this one.
- \$6.99 [HDMI cable](#)
- Monitor/TV with hdmi
- SD Card, 8GB minimum, 32GB maximum

Advantages:

- Full Linux like OS based on debian
- Good environment for learning Linux and Python
- Reasonable interfaces to IoT sensors
- excellent camera support
- excellent choice of expansion packages including Grove Sensors that make it easy to attach sensors and actuators. An example package is the [Grove Starter Kit](#) for about \$90

Disadvantages:

- We tried the Windows IoT package and were not impressed by it. This is not an issue of the Raspberry, but the Windows IoT platform

72.2 ESP8266 Robot Car Kit

The ESP8266 has many variants. Some of which are difficult to interface with. However, this does not apply for the ESP8266 NodeMCU. This board is originally flashed with *Lua*, however it can easily be reflashed with MicroPython. In addition it is often offered as part of a platform to develop a robot car. There are arguably better kits available, but the price of \$24 for the entire kit is hard to beat. Unfortunately the version of python, as well as the limited memory make the esp8266 not a full fledged platform for python programming and you will quickly see its limitations. Interfacing with it, however, as an IoT device will gain you a lot of insights.

Configuration:

- \$14.99 [esp8266 & shield](#) or [esp8266 & shield](#)
- \$12.59 [Chasis](#)
- 4 * AA Rechargeable Batteries & charger

Optionally you may want to get additional sensors such as wheel Encoders

- [Wheel Encoder](#)

Advantages:

- Very low price for what it can do
- We have OSX software available that makes it easy to setup (Other tutorials for other platforms are available on the internet, you can contribute by creating documentation we distribute in class for points)
-
-

72.3 Sensor Kit

It is fun to attach sensors to your IoT board. There are many kits available and we encourage you to do comparisons. One such kit is

- \$29.99 [Elegoo 37 Sensors](#)

However it does not include a breadboard like other kits. Hence we recommend that you get a breadboard as it makes experimenting easier.

- \$5.68 [small bread board and wires](#)

72.4 Fish Kit

- \$29.99 [shark](#)
 - cheaper balloons leak
 - before assembly and putting gas in, make sure components work.
 - gas will last typically for one week
 - \$39.99 gas can be purchased in party store
- 2 g9 servo
- soldering (for cable, so cheap one will do)
- pins
- esp
- double sided scotch tape
- hot glue gun
- paper clips

72.5 Alternative components

72.5.1 Esp8266 Alternatives

Two models are good. Adafruit has some added features, but may need soldering

- \$8.79 [NodeMCU](#)
- \$16.95 [Adafruit Feather](#)

72.5.2 Car Parts Alternatives

- \$14.59 [Car Chasis](#)
- \$22.88 [Car Chasis and Arduino](#)

72.5.3 Simple sensors

Simple sensors can be attached to the boards with cables (that you need to purchase separately). Examples include

- [Elegoo 37 sensor kit](#)
- [Breadboard Cable](#)

72.5.4 Grove Sensors

Grove sensors have ready-made cables that make them easy to attach to the Raspberry Pi. However, they are more expensive. You still need a Raspberry Pi. No soldering iron and no breadboards are required.

- [Grove Starter Set](#)
- [Seed Studio Grove Sensors](#)
- [Grove Shield for NodeMCU](#)
- [Grove Cable](#)

72.6 Alternative Hardware and Sensors

In this section we will list a number of alternative hardware products that we are exploring. If you have used them, please help us improving these sections.

72.6.1 Small Footprint Battery Power

The following board provides to the Raspberry Pi a lithium battery power pack expansion board. It is powered by two 18650 Li-ion batteries providing steady. A 4-LED indicator indicates the level of charge. The board costs \$16.99.

- <https://www.sunfounder.com/plus-power-module.html>



73. Projects

Please see the introduction to the IoT section to get started.

Term project suggestion combining IoT and Big Data:

1. Recognizing street sign in a car robot with a camera
2. Recognizing street lines in a car robot with camera
3. Driving a Robot car swarm without collisions
4. Simulating a City with robot cars
5. Control a robot fish with cameras
6. Build a distributed sensor system (with your classmates)

Drones:

1. Control a drone swarm with positioning system

Suggest your own



74. ESP8266

When working with an external hardware such as the NodeMCU you will find a lot of information on the internet about it. It is a bit difficult at times to assess what you need to program it. You are exposed to many choices. A NodeMCU typically comes with Lua. However you have many other choices. Such choices include multiple programming languages such as Lua, MicroPython, Arduino/C, Go and others.

As all of them are slightly different you need to identify which works best for you. In addition you need to install images, programs and libraries that support your specific language choice.

For our first experiments we will be using MicroPython. This choice is motivated by the fact that Python is a well established and easy to learn programming language. Recently many educational institutions are offering Python as an introductory programming language making this choice even more compelling.

To simplify the setup and use of the esp8266 for MicroPython we developed an easy to use commandline tool that allows users to set up their computer and interact more easily with the board. We believe that the interface is so simple that it can also be used in STEM activities and not just in the university or by advanced hobbyists.

74.0.1 Installation

In this section we discuss the various ways on how to set up the esp8266 `cloudmesh.robot` development environment. You have several options to install it.

1. Option A: OSX with scripts hosted on github (recommended)
2. Option B: OSX from source
3. Option C: Explore your own

While we provide here a detailed option for OSX, you are free to explore other operating systems.

We know that it can for example be installed on Ubuntu 16.04. We have not tested any of this on a Windows machine.

We like to get feedback and installation instructions.

Option A: OSX install from a Script

For OSX we have created two scripts that you will need

- [system.sh](#), that installs pip, ansible, homebrew, xcode, virtualenv, readline, wget, lua, picocom, mosquito, aquamacs, pycharm, numpy, matplotlib, libusb, USB drivers for selected esp8266 (ch34x chip)
- [user.sh](#), that installs matplotlib, virtualenv, and the cloudmesh source in ~/github

We recommend that you review these scripts carefully before you use them and check if they fit your needs. If they do not, please just download them and adapt them to your needs. The **system** script must be ran on an **Administrator** account as it requires sudo privileges. The **user** script must be ran on a **User** account. We do not recommend to run the IoT software in an administrative account due to security best practices. To execute the **system** script, type in the *Administrator account* terminal

```
curl -fsSL http://cloudmesh.github.io/get/robot/osx/system | sh
```

As earlier versions of pip may have some issues, this script will also update pip and setuptools to a newer version

To execute the **user** script, type in the User account terminal

```
curl -fsSL http://cloudmesh.github.io/get/robot/osx/user | sh
```

Together these scripts allow you to install in a simple way development tools for our IoT activities.

The following steps are to be executed in the user environment.

Warning: *the scripts do not update pip and setuptool, which may be required due to a bug in setuptools prior to version 34 for setuptools. You may have to repeat the update on any pyenv environment that you use. How to do this is documented in a later section.*

To simplify use, we recommend that you make the following additions to your ~/.bash_profile file so that python 3 is automatically activated, but does not interfere with the system installed python. Use the command

```
$ emacs ~/.bash_profile
```

or your favourite editor to edit the file and add the following lines at the end.

```
#####
# PYENV
#####
open_emacs() {
    # open -na Aquamacs $*
    open -a Aquamacs $*
}
alias e=open_emacs

#####
# PYENV
#####
export PYENV_VIRTUALENV_DISABLE_PROMPT=0
eval "$(pyenv init -)"
```

```
eval "$(pyenv virtualenv-init -)"
__pyenv_version_ps1() {
  local ret=$?;
  output=$(pyenv version-name)
  if [[ ! -z $output ]]; then
    echo -n "($output)"
  fi
  return $ret;
}
PS1="\$__pyenv_version_ps1) ${PS1}"
alias ENV3="pyenv activate ENV3"
ENV3
```

Once you start a new terminal you can edit files via aquamacs by typing

```
e FILENAME
```

where FILENAME is the name of the file you like to edit. However the file must exist, which you can simply do with

```
touch FILENAME
```

Add the following lines at the end of the file

To learn more about how to you automate the setup of an OSX machine, you may be inspired by

- https://github.com/ricbra/dotfiles/blob/master/bin/setup_osx
- <https://blog.vandenbrand.org/2016/01/04/how-to-automate-your-mac-os-x-setup-with-ansible/>

Setting Up Git

Sooner or later you will be using git. We recommend that you set your identity on all computers that you will be using. To do this adapt the following example according to your github.com identity that you have. IF you do not, its time to create one at github.com and follow the directions.

```
$ git config --global user.name "Gregor von Laszewski"
$ git config --global user.email laszewski@gmail.com
$ git config --global core.editor emacs
$ git config --global push.default matching
```

74.0.2 Option B: setup from pip

We have removed the pip setup instructions as they do not include installing the drivers.

Option B: Install Cloudmesh Robot from source

Developers that already have a development environment (e.g. xcode is installed) can install cloudmesh robot also from the terminal while downloading the source. You will need to first obtain the source and compile it with the following commands:

```
$ mkdir github
$ cd github
$ git clone https://github.com/cloudmesh/cloudmesh.common.git
$ git clone https://github.com/cloudmesh/cloudmesh.cmd5.git
$ git clone https://github.com/cloudmesh/cloudmesh.robot.git
$ cd cloudmesh.robot
$ make source
```

To test out if the command has been installed, type

```
$ cms robot welcome
```

If everything works you should see an ASCII image of R2D2 and C3PO. Next, we still have to install some additional programs before you can use other commands.

Once you have installed cloudmesh robots you will be able to install a number of tools automatically with the command

```
$ cms robot osx install
```

This will install services and tools including xcode, homebrew, macdown, pycharm, and aquaemacs. If you have some of these tools already installed it will skip the installation process for a particular tool. Please note that some of the tools require root access and thus you must be able to have access to sudo to run them from our tool. In addition you will need to install the OSX driver for the USB interface to the esp8266. This is achieved with (only to be done if you follow the install from source option)

```
$ cms robot osx driver
```

Now please change your account to be again a standard account.

Now you **MUST REBOOT** the machine. Without rebooting you will not be able to use the USB drivers.

74.0.3 Option C: A possible setup for Linux

On a linux computer we recommend that you install emacs, cmake and configure your git. Replace the user name and e-mail with the one that you used to register your account in git:

```
$ mkdir github
$ cd github
$ git clone https://github.com/cloudmesh/cloudmesh.robot.git
$ ssh-keygen
$ sudo apt-get install -y emacs
$ sudo apt-get install -y cmake
$ sudo apt-get install -y libqt4-dev
$ git config --global user.name "Gregor von Laszewski"
$ git config --global user.email laszewski@gmail.com
$ git config --global core.editor emacs
$ git config --global push.default matching
```

This setup is highly incomplete and does not include the setup of the USB drivers. Please help us completing the documentation.

Option C: A possible setup for Windows

We do not have tried to set this up on Windows or a virtualbox running Linux under windows. If you have tried it, please let us know. If you have difficulties just use a raspberry PI and skip the IoT projects

Option C: Installation of the cloudmesh.robot Interface via Pip

.. warning:: this option does not include installing the USB drivers. You have to install them first. See examples on how to do that in our install scripts. Generally what we do in our user.sh script is the same way, but also includes the setup of python 3.6.1.

To more easily interface with the robot we have developed a convenient program that is installed as part of a command tool called cloudmesh.

Install Cloudmesh Robot with Pip (not working)

Note that pip may not include the newest version of cloudmesh.robot and we recommend you use the source install instead.

```
$ pip install cloudmesh.robot
```

This will install a program cms on your computer that allows you to easily communicate with the robot.

74.0.4 Using cloudmesh robot

Once you have successfully installed the drivers and the commands you can look at the manual page of the robot command with

```
$ cms robot help
```

You will see a manual page like this:

```
Usage:
robot welcome
robot osx install
robot osx driver
robot image fetch
robot probe [--format=FORMAT]
robot flash erase [--dryrun]
robot flash python [--dryrun]
robot test
robot run PROGRAM
robot credentials set SSID USERNAME PASSWORD
robot credentials put
robot credentials list
robot login
robot set PORT NOT IMPLEMENTED
robot ls [PATH]
robot put [-o] SOURCE [DESTINATION]
robot get PATH
robot rm PATH
robot rmdir PATH
robot dance FILE IPS
robot inventory list [--cat] [--path=PATH] [ID]
```

Testing the board

Before you can use you ESP8266, you must have the appropriate drivers installed on your computer. Click on [this link](#) and follow the instructions on how to install these drivers.

Next is to connect a esp8266 with a USB cable to the computer. The ESP8266 should look similar to this.

After you connected it, press the reset button. Before doing anything on the board, we must test it. Once you have plugged it in, execute the following command:

```
$ cms robot probe
```

This command takes about ten seconds to execute. The ESP8266's led should flash irregularly as it is probed. When the probe is finished, an image similar to the following should appear in your terminal:

```
+-----+-----+
| Attribute | Value |
```

```
+-----+-----+
| chipid | b' 0x00d0f9ec' |
| mac    | b' 00:10:FA:6E:38:4A' |
| tty    | /dev/tty.wchusbserial1410 |
+-----+-----+
```

Please note that you should only have one board attached to your computer.

Flashing the image onto the robot board

Next we need to flash the image on the robot board. Naturally we need to fetch the image first from the internet. We do this with the command

```
$ cms robot image fetch
```

This will fetch an image that contains MicroPython into your local directory.

Next we need to *flash* the image on the board.

Before you begin, make sure that the ESP8266 is connected to your computer. The board may come with a preinstalled image such as Lua or some custom image from the vendor. In order to write programs in python, we need to the chips to run micropython. To get micropython on our ESP8266's, a number of steps are required.

Erase the chip

First we need to erase the chip.

Run the following command in your terminal terminal, and then **stop**.

```
$ cms robot flash erase
```

Your terminal should respond with the following query:

```
/dev/tty.SLAB_USBtoUART
Please press the right buttons
continue? (Y/n)
```

Before taking any further steps, press both buttons on the ESP8266 at the same time. Once you have done this, type Y and press enter. The process should take under ten seconds to complete.

Putting Python on the chip

Before proceeding, you must once again press both of the buttons on the ESP8266. Once this is done, you are ready to flash the chip with python with the following command:

```
$ cms robot flash python
```

Testing if it works

To test running a pyton program execute

```
$ cms robot test
```

Be careful as it overwrites the file `test.py`. If the ESP8266 is set up properly, it should return this in your terminal:

```
Count to 3
1
2
3
```

Execute an arbitrary program

Lets assume you have placed a program in the file `prg.py` with the command

```
$ cms robot put prg.py
```

You must reboot the ESP8266 before using a new program. This can be done manually by pressing the reset button on the chip, or in terminal with the command

```
$ cms robot reset
```

Once the chip is reset, you can run `prg.py` with the following command:

```
$ cms robot run prg.py
```

Interactive Python shell on the board

To get into the interactive python shell on the board you need to reset the ESP8266 and run the following command:

```
$ cms robot login
```

Cleaning an reinstalling a development version

IN case you are a developer and you need to modify the source code, we found that it is sometimes necessary to clean your development directory and libraries. The easiest way to do this is to go to the repository that you like to reinstall. Let us assume it is *cloudmesh.robot*. Than the following commands will clean the repository

```
$ cd cloudmesh.robot
$ pip uninstall cloudmesh.robot
```

Do the pip unisntall as many times till you see an error that no more cloudmesh.robot versions can be found. Than execute

```
$ make clean
```

After this you can reinstall it with

```
$ python setup.py install; pip install -e .
```

the `-e` flag is optional, but allows you to change the code without the need of recompiling. A very useful feature in python.

NodeMCU ESP12 Dev Kit Pin Definition

For V1.0

The GPIO numbers of teh NodeMCU, do not correspond with the actual numbers used in micropythons pin library. The numbers are as follows:

Pin/GPIO	NodeMCU
15	D8

LED

Program

```
import machine
led = machine.Pin(15,machine.Pin.OUT)
led.high()
```

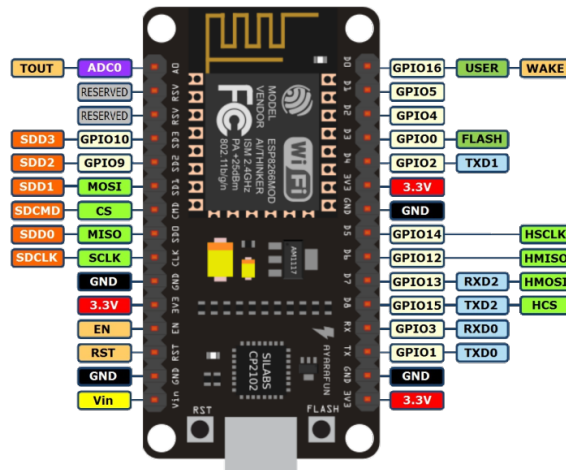


Figure 74.1: nodemcu

```
led.low()

import machine
led = machine.Pin(15,machine.Pin.OUT)
while True:
    led.high()
    time.sleep(0.5)
    led.low()
    time.sleep(0.5)
```

Real Time Clock

Get the library `urtc.py`:

```
wget https://raw.githubusercontent.com/adafruit/Adafruit-uRTC/master/urtc.py
```

Place it on the esp8266

```
cms robot put urtc.py
```

Connect the board the following pins

```
SDA to pin 5 = D1
SCL to pin 4 = D2
```

Login to the board

```
cms robot login
```

Execute the following code

```
import machine
i2c = machine.I2C(sda=machine.Pin(5), scl=machine.Pin(4))
i2c.scan()

[87, 104]

from urtc import DS3231
t = DS3231(i2c)

t.datetime()
```

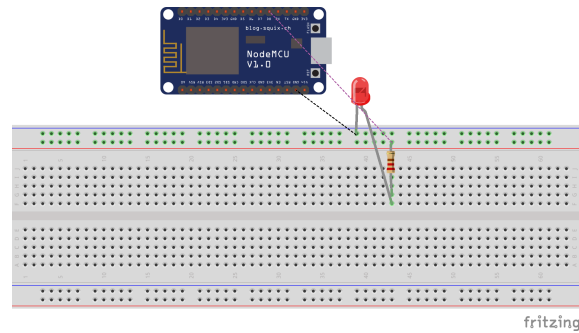


Figure 74.2: breadboard

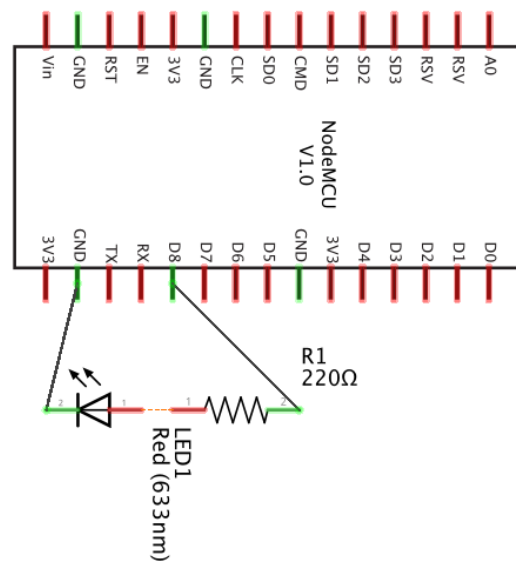


Figure 74.3: schema

```

DateTimeTuple(year=2000, month=1, day=1,
              weekday=1, hour=0, minute=15,
              second=53, millisecond=None)

```

Assignment: Create an object oriented class and fill out the details while using code from `urtc.py`

```

class Clock (object):
    def __init__(self, sda=5, scl=4)
    def get(self)
    def __str__(self)

c = Clock()
print (c.get())
print (c)

```

74.0.5 Resources

<https://github.com/adafruit/Adafruit-uRTC/blob/master/urtc.py>
 git clone <https://github.com/adafruit/Adafruit-uRTC.git>

74.0.6 Alternative boards

HUZZAH Feather esp8266

Many different 8266 based alternative boards exist. One of these boards is the HUZAZH Feather. IT behaves the same as the other boards, but may be using different drivers and USB ports. The *cms robot* command line tool is clever enough to identify automatically if it is attached and uses the appropriate settings. More documentation about this board can be found at

- [doc](#)

This site has also many other examples and you can search for them with keywords such as feather, esp8266, micropython.

An example on how to use the LED on the *feather* is documented at

- [Feather HUZAZH ESP8266](#)

To place micropython on the feather you can plug in the to the usb port. The good thing about this board is that you do not need to press any buttons as it detects the upload nicely. If not make sure to reset it or for flashing press both buttons. You can do the following:

Probe the board with:

```
cms robot probe
```

Erasing the feather is simple as it has a build in mechanism to detect if it is going to be erased. Hence no reset button needs to be pressed:

```
cms robot flash erase
```

Get the python image:

```
cms robot fetch python
```

Flashing is conducted with 460800 baud, it will take about 15 seconds. After flashing you should try to login:

```
cms robot login
```

Set the baudrate to 115200:

```
CTRL-A CTRL-B>
```

```
*** baud:
```

type in:

```
115200 <ENTER>
```

Make sure that echo is switched to OFF:

```
CTRL-A CTRL-C
```

toogles it. Now you should see:

```
>>>
```

Try typing in:

```
print("Hello")
```

74.0.7 Appendix

74.0.8 Installing ESP8266 USB drivers

We provide here a section to explain which drivers we have tested on various esp8266. Please note that if you have different versions you may need different drivers. On OSX we found that we get good results with the following commands

```
brew tap mengbo/ch340g-ch34g-ch34x-mac-os-x-driver https://github.com/mengbo/ch340g-ch34g-ch34x-mac-o
brew cask install wch-ch34x-usb-serial-driver
```

Start a new terminal after the driver has finished installing.

```
wget http://www.silabs.com/Support%20Documents/Software/Mac_OSX_VCP_Driver.zip
unzip Mac_OSX_VCP_Driver.zip
```

Click on the driver install file contained inside the zip file and the driver should start installing. Once the driver has finished installing, make sure to start a new terminal.

Please remember that you need to close all terminals, as well as reboot the computer to use the drivers. They will typically not work if you have not rebooted.

For other boards that also use the CH340G chip the following page may help:

- <http://kig.re/2014/12/31/how-to-use-arduino-nano-mini-pro-with-CH340G-on-mac-osx-yosemite.html>

75. Raspberry PI 3

75.1 Raspberry PI for IOT (Gregor)

75.2 Hardware

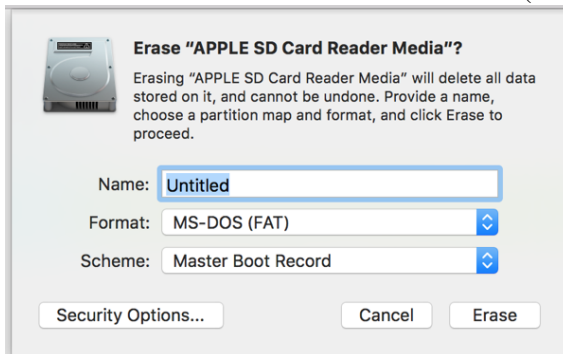
see hardware page we have

75.3 Installation

75.3.1 Erasing the SD Card

Before you can install an OS on your sd card, you must erase it and put it in the proper format.

1. Insert your sd card into your micro-sd adapter and open Disk Utility with a spotlight search.
2. In the Disk Utility, right click the name of the sd card and select erase.
3. Name the sd card and format it as MS-DOS (FAT). Then click erase.



4. If it does not erase the first time, try again. It sometimes takes multiple tries to work.

75.3.2 Installation of NOOBS

NOOBS is an OS that includes Raspbian. The official description of Raspbian can be found [here](#). It comes pre-packaged with many useful programming tools, and is easy to use.

1. Download Noobs [here](#). This will take around 30 minutes.
2. Go to your Finder and in Downloads, search for NOOBS.
3. Open the NOOBS folder and drag its contents into the sd card in the devices section. There should be 20 files and folders in the NOOBS folder. The download should take about 3 minutes.
4. Once installed, eject the sd card and put it in your raspberry pi.
5. Power up your raspberry and you will see a menu like this

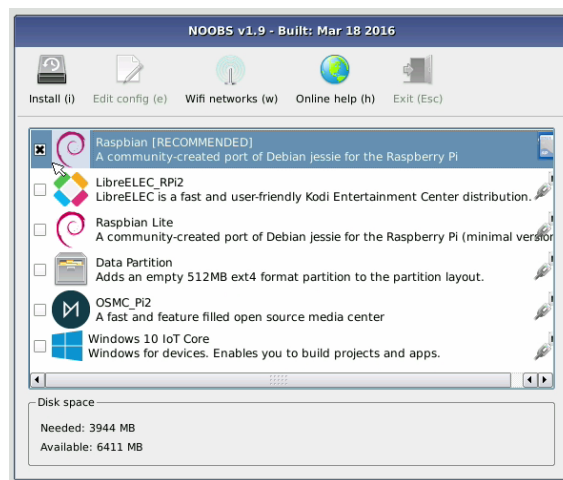


Figure 75.1: Noobs

6. Select Raspbian and click Install (i)

75.3.3 Installation of Dexter

The version of Dexter that you want to flash onto your sd card is called Raspbian for Robots. This is a Raspbian based os that is compatible with the GrovePi board. It also comes with pre-installed Dexter Industries software.

1. First, download the most recent Dexter_Industries_jessie.zip file from [here](#).
2. Once the file has downloaded, uncompress it and insert your sd card into the micro-sd adapter.
3. Open etcher and flash the uncompressed jessie image onto the sd card.
4. Eject your sd card and insert it into your raspberry pi.

75.4 Configure

75.4.1 Prepare OS

75.5 Update

The following are essential updates:

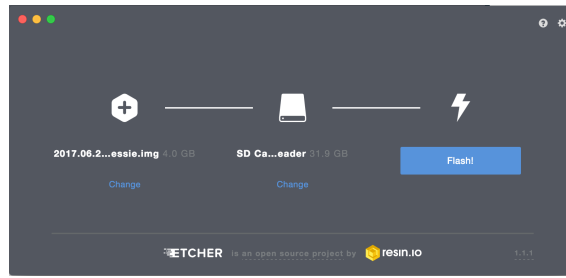


Figure 75.2: Etcher

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install emacs
dpkg -l > ~/Desktop/packages.list
pip freeze > ~/Desktop/pip-freeze-initial.list
```

The following are necessary for the scientific libraries, but they require lots of space. Our sd cards do not have enough space for them.

```
sudo apt-get install build-essential python-dev python-distlib python-setuptools python-pip python-wh
sudo apt-get install xsel xclip libxml2-dev libxslt-dev python-lxml python-h5py python-numexpr python
sudo pip install bottleneck rtree
```

add to .bashrc

```
cd
git clone git://github.com/yyuu/pyenv.git .pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(pyenv init -)'" >> ~/.bashrc
source ~/.bashrc
```

```
export PATH="/home/pi/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

```
curl -L https://raw.githubusercontent.com/pyenv/pyenv-installer/master/bin/pyenv-installer | bash
```

source

75.5.1 Update to Python 3.6.1

75.6 change python version

- [<https://linuxconfig.org/how-to-change-from-default-to-alternative-python-version-on-debian-linux>] (<https://linuxconfig.org/how-to-change-from-default-to-alternative-python-version-on-debian-linux>)

Upgrade setuptools for pip install with

```
$ pip3 install --upgrade setuptools
```

Test your python version with

```
$ python --version
```

Check your python version alternatives

```
$ update-alternatives --list python
```

If python2.7 is not in your alternatives, add it with

```
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
```

If python3.4 is not in your alternatives, add it with

```
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.4 2
```

Now make python3.4 to your default with

```
update-alternatives --config python
```

Select python3.4

75.7 install 3.6.1

To install python 3.6.1, follow steps 1 and 2. This is unnecessary for our purposes.

- [better get 3.6.1](#)

75.8 install cloudmesh.pi

```
pip install cloudmesh.pi
```

pip install cloudmesh.pi with

```
$ git clone https://github.com/cloudmesh/cloudmesh.pi.git
$ cd cloudmesh.pi
$ sudo pip3 install .
```

see how we do this in osx/linux can this be done on raspbery? if not document update from source with altinstall

75.8.1 Install scientific Libraries

check if they are already installed we don't have enough space to install all of these.

```
sudo apt-get install python-numpy python-matplotlib python-scipy python-sklearn python-pandas
```

```
numpy
matplotlib
scipy
scikitlearn
```

75.8.2 cloudmesh.pi (Jon)

cloudmesh.pi is a repository for our GrovePi module classes. These classes require Dexter software, so you need to either have Raspian for Robots or download the software separately.

If you have Raspian for Robots, run the following in your terminal:

```
cd
mkdir github
cd github
```

```
git clone https://github.com/cloudmesh/cloudmesh.pi.git
cd cloudmesh.pi
sudo pip install .
```

75.8.3 Install VNC

describe how to install and configure VNC

75.9 Sensors (Jon)

75.9.1 Grove Sensors (Jon)

we already have draft

75.9.2 Non Grove Sensors (Jon)

Elegoo as example

75.10 Notes To integrates

75.10.1 Connecting

Hostnames:

- raspberrypi.local
- raspberrypi.

change

recovery.cmdline

forgot what these were:

```
runinstaller quiet ramdisk_size=32768 root=/dev/ram0 init=/init vt.cur_default=1 elevator=deadline
silentinstall runinstaller quiet ramdisk_size=32768 root=/dev/ram0 init=/init vt.cur_default=1 elevat
```

Connect the cable

You will see the activity LEDs flash while the OS installs. Depending on your SD-Card this can take up to 40-60 minutes.

75.11 VLC on OSX

- http://www.videolan.org/vlc/index.en_GB.html
- <http://get.videolan.org/vlc/2.2.6/macosx/vlc-2.2.6.dmg>
- <http://www.mybigideas.co.uk/RPi/RPiCamera/>
-

75.12 Camera on Pi

sudo apt-get install vlc

- <https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/>
- <https://www.hackster.io/bestd25/pi-car-016e66>

75.13 Streaming video

- <https://blog.miguelgrinberg.com/post/stream-video-from-the-raspberry-pi-camera-to-web-browsers-even-on-ios-and-android>

75.14 Linux Commandline

- <http://www.computerworld.com/article/2598082/linux/linux-linux-command-line-cheat-sheet.html>

75.15 Enable SPI

go to the configuration interfaces and enable

75.16 RTIMULib2

git clone <https://github.com/RTIMULib/RTIMULib2.git> cd RTIMULib

Add the following two lines to /etc/modules

```
i2c-bcm2708
i2c-dev
```

reboot

```
ls /dev/i2c-*
sudo apt-get install i2c-tools

sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  68  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

create a file /etc/udev/rules.d/90-i2c.rules and add the line:

```
KERNEL=="i2c-[0-7]",MODE="0666"
```

note: does not work

instead we do

```
sudo chmod 666 /dev/i2c-1
```

Set the I2C bus speed to 400KHz by adding to /boot/config.txt:

```
dtparam=i2c1_baudrate=400000
```

reboot. In terminal change directories to

```
cd /home/pi/github/RTIMULib2/RTIMULib/IMUDrivers
```

and open

```
emacs RTIMUdefs.h
```


Raspberry Pi 3 GPIO Header				
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

Figure 75.3: Pinout

In RTIMUdefs.h change

```
#define MPU9250_ID 0x71
```

To

```
#define MPU9250_ID 0x73
```

```
cd /home/pi/github/RTIMULib2/RTIMULib
```

In terminal

```
mkdir build
cd build
cmake ..
make -j4
sudo make install
sudo ldconfig
```

75.17 Compile RTIMULib Apps

```
cd /home/pi/github/RTIMULib2/Linux/RTIMULibCal
make clean; make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDrive
make clean; make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDrive10
make clean; make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDrive11
make clean; make -j4
sudo make install
```

```

cd /home/pi/github/RTIMULib2/Linux/RTIMULibDemo
qmake clean
make clean
qmake
make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDemoGL
qmake clean
make clean
qmake
make -j4
sudo make install

```

75.18 Camera

- [Camera Tutorial](#)

```

sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev

sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip

```

copy into ~/.profile:

```

echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile

```

source ~/.profile

```
mkvirtualenv cv -p python3
```

workon cv

comandline has (cv) in front

```

pip install numpy

wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
unzip opencv.zip
unzip opencv_contrib.zip

```

75.19 Lessons and Projects

- [Gui](#)
- [Solder](#)
- [PI Camera Line Follower](#)
- [Pi car flask](#)

75.20 OTHER TO BE INTEGRATED

75.20.1 PI emulator on Windows

We have not yet tried it, but we like to hear back from you on experiences with

- <https://sourceforge.net/projects/rpiqemuwindows/>

75.20.2 Scratch

- [scratch](#)

75.21 Web Server

- [Web Server Flask](#)



76. Dexter

76.1 Creating an SD Card

76.1.1 OSX

First, install Etcher from etcher.io which allows you to flash images onto the SD card. When flashing make sure you only attach one USB SD card reader/writer or use the build in SD card slot provided in some Mac's.

The version of etcher we used is

- [Etcher-1.1.1-darwin-x64.dmg](#)

Make sure to check if there is a newer version

76.1.2 Dexter Rasbian

Dexter provides a special image that contains the drivers and sample programs for the GrovePi shield. We had some issues installing it on a plain Raspbian OS, thus we recommend that you use dexters version if you use the GrovePi shield. It is available from

- [Google Drive](#)
- [Sourceforge](#)

Detailed information on how to generate an SD card while using your OS is provided at

- <https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card/>

76.1.3 Github

Dexter maintains a github repository that includes their code for the shield and many other projects at

- <https://github.com/DexterInd>

76.1.4 Cloning Grove PI

To clone the GrovePI library on other computers you can use the command

```
git clone https://github.com/DexterInd/GrovePi.git
```

76.1.5 Dexter Sample programs

Dexter maintains all GrovePi related programs at

- <https://github.com/DexterInd/GrovePi>

The python related programs are in a subdirectory at

- <https://github.com/DexterInd/GrovePi/tree/master/Software/Python>

Here you find many programs and for a complete list visit that link. Dependent on the sensors and actuators you have, inspect some programs. Some of them may inspire you to purchase some sensors.

We have developed a partial library of GrovePi module classes at

- <https://github.com/cloudmesh/cloudmesh.pi/tree/master/cloudmesh/pi>



77. GrovePi Modules

77.1 Introduction

- [Electronics](#): An introduction to the basic principals of electronics.
- [Volatage](#): An introduction to the physics of electricity.
- [Unix](#): An introduction to the Unix os.
- [grove examples](#): A list of Dexter Industries example code for GrovePi modules.
- [GrovePi module classes](#): A repository for the GrovePi module classes.

77.2 LED

An LED is the simplest possible module for a raspberry pi, as it is responsive only to the provided power. For an LED to emit light, it must be exposed to a voltage greater than a certain threshold value. Above this voltage, the conductivity of the diode increases exponentially and its brightness increases likewise. If the current through the LED becomes too high, the LED will burn out. The following link leads to a tutorial from Dexter Industries for the LED module.

- [Dexter LED tutorial](#)

Connect the LED To a digital port. The following code describes an LED class. Since it is connected to a digital output, the voltage has only two states, on and off. The default port for the LED class is D3. The code for the LED class can be found here:

- [LED Class](#)



Figure 77.1: LED



Figure 77.2: Buzzer

77.3 Buzzer

Connect the buzzer to a digital port. The default port for the Buzzer class is D3. You will notice that the Buzzer class and the LED class are interchangeable. This is because they work on the same digital principal. Their two values are on and off. The code for the Buzzer class can be found here:

- [Buzzer Class](#)

77.4 Relay

The relay acts as a switch in a circuit. When the value on the relay is 1, it allows current to flow through it. When the value is 0, the relay breaks the circuit and the current stops. Connect the relay to a digital port. The default digital port is D4. The Relay class can be found here:

- [Relay Class](#)



Figure 77.3: Relay



Figure 77.4: Light Sensor

77.5 Light Sensor

The light sensor measures light intensity and returns a value between 0 and 1023. Connect the light sensor to an analog port. The default port is A0. The analog port allows the light sensor to return a range of values. The `LightSensor` class can be found here:

- [LightSensor Class](#)

77.6 Rotary Angle Sensor

The rotary angle sensor measures the angle to which it is turned. Connect the sensor to an analog port. Port A0 is the default. The `RotarySensor` class can be found here:

- [RotarySensor Class](#)

77.7 Barometer

Connect the barometer to an I2C port. In addition to pressure, the GrovePi barometer measures temperature in Fahrenheit and Celsius. The `Barometer` class can be found here.



Figure 77.5: Rotary Angle Sensor

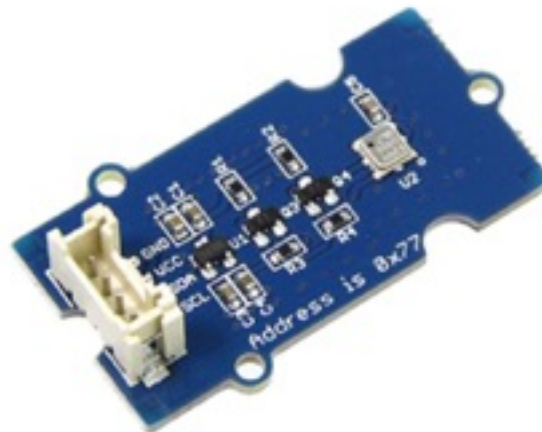


Figure 77.6: Barometer

- [Barometer Class](#)

77.8 Distance Sensor

Connect the distance sensor to a digital port. The grovepi module has a built-in function to read the distance from the distance sensor, but it is improperly calibrated, so this DistanceSensor class has a calibration based on experimental data. The DistanceSensor class can be found here:

- [DistanceSensor Class](#)

77.9 Temperature Sensor

The temperature sensor measures both temperature and humidity. Connect the temperature sensor to a digital port. D7 is the default port. The TemperatureSensor class can be found here:



Figure 77.7: Distance Sensor

- [TemperatureSensor Class](#)

77.10 Heartbeat Sensor

Connect the heartbeat sensor to an I2C port. The heartbeat sensor returns the heart rate of the wearer. The HeartbeatSensor class can be found here:

- [HeartbeatSensor Class](#)

77.11 Joystick

Connect the joystick to an analog port. A0 is the default port. The joystick has an x, y, and click status based on the current state of the module. The Joystick class can be found here:

- [Joystick Class](#)

77.12 LCD Screen

The LCD screen can be used to display text and colors. In order to use it, plug it into one of the I2C ports. The LCD class can be found here:

- [LCD Class](#)

77.13 Moisture Sensor

Connect the moisture sensor to an analog port. The default port is A0. The MoistureSensor class can be found here:

- [MoistureSensor Class](#)

An example of the implementation of the moisture sensor from Dexter Industries can be found [here](#). The program is meant to measure the environmental conditions that affect plant growth.



Figure 77.8: Temperature Sensor



Figure 77.9: image



Figure 77.10: image



Figure 77.11: LCD Screen

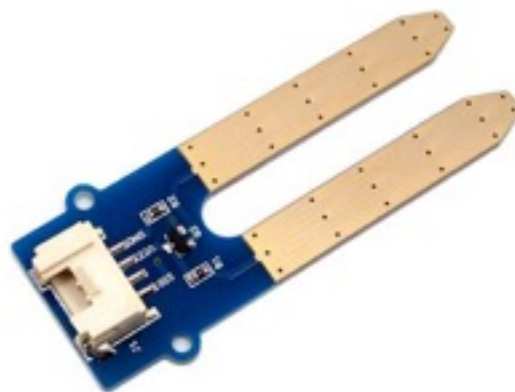


Figure 77.12: Moisture Sensor



Figure 77.13: Water Sensor

77.14 Water Sensor

The water sensor measures the amount of water in the environment of the sensor. Connect the sensor to a digital point. D2 is the default port. The `WaterSensor` class can be found here:

- [WaterSensor Class](#)

77.15 Sensors

This section is to be completed by the students of the class.

Task is to develop an object oriented class for one of the sensors. An example for such a class can be found at:

- <https://github.com/cloudmesh/cloudmesh.pi/blob/master/cloudmesh/pi/led.py>

77.15.1 Compass

TODO: which compas sensor

The default pins are defined in `variants/nodemcu/pins_arduino.h` as GPIO

```
SDA=4
SCL=5
D1=5
D2=4.
```

You can also choose the pins yourself using the I2C constructor `Wire.begin(int sda, int scl);`



78. VNC

Note: *If you like to connect to your Raspberry from your laptop, we recommend to use VNC. If you rather like to connect a monitor and keyboard as well as a mouse to the Raspberry, you can skip the steps with the VNC update.*

78.1 Setting up VNC

We had some issues with the installed version of VNC that is customized for connecting a Laptop via the ethernet cable to the PI. However as we connect wirelessly, our setup is slightly different. The easiest way that we found is to update the Raspbian OS as follows. In a terminal type

```
sudo apt-get update
sudo apt-get install realvnc-vnc-server
sudo apt-get install realvnc-vnc-viewer
```

Next you enable the VNC server in the configuration panel via the Raspbian GUI by selecting

```
Menu >
  Preferences >
    Raspberry Pi Configuration >
      Interfaces.
```

Here you toggle the VNC service to enabled. As we are already at it in our setup we enabled all other services, especially those that deal with Grove sensor related bins and wires.

Next reboot and double check if the settings are preserved after the reboot

78.1.1 Install VNC on OSX

To install a vnc server of your liking on your Mac. You find one at

- <http://www.realvnc.com/download/vnc/latest/>

Be sure to download the version of the VNC Viewer for the computer you are going to use to virtually control the Pi (there is a version listed for Raspberry Pi-- don't download this one. For us this is the Mac version.)

78.1.2 Run VNC Viewer on OSX

Once you have downloaded the VNC viewer installed it you can open the program. Next you can start vnc viewer and enter the ip address of your raspberry. Make sure you are on the same network. You can find the address by using ifconfig.



79. Turtle Graphics

79.1 Demo

Python comes with a nice demonstration program that allows you to learn some simple programming concepts while moving a turtle on the screen. It can be started with

```
python -m turtledemo
```

79.2 Program example

You can also create programs with your favorite editor and run it. Let us put the following code into the program `turtle_demo.py`. Never save a file with the name `turtle.py` because python will import it instead of the built-in turtle import that you need.

```
import turtle

window = turtle.Screen()
robot = turtle.Turtle()

robot.forward(50)    # Moves forward 50 pixels
robot.right(90)     # Rotate clockwise by 90 degrees

robot.forward(50)
robot.right(90)

robot.forward(50)
robot.right(90)

robot.forward(50)
robot.right(90)
```

```
turtle.done()

window.mainloop()
```

After saving it you can run it from a terminal with

```
$ python turtle_demo.py
```

79.3 Shape

```
shapes: "arrow", "turtle", "circle", "square", "triangle", "classic"
```

You can change the shape of your turtle to any of these shapes with the Turtle method `shape(name)`. For example, if you have an instance of the Turtle class called `robot`, you can make it appear as a turtle by calling `robot.shape("turtle")`.

You can add your own shapes with the following functions:

```
turtle.register_shape(name, shape=None)

turtle.addshape(name, shape=None)
```

There are three different ways to call this function:

`name` is the name of a gif-file and `shape` is `None`: Install the corresponding image shape.

```
window.register_shape("turtle.gif")
```

Note: Image shapes do not rotate when turning the turtle, so they do not display the heading of the turtle!

`name` is an arbitrary string and `shape` is a tuple of pairs of coordinates: Install the corresponding polygon shape.

```
window.register_shape("triangle", ((5,-3), (0,5), (-5,-3)))
```

`name` is an arbitrary string and `shape` is a (compound) Shape object: Install the corresponding compound shape.

Add a turtle shape to TurtleScreen's `shapelist`. Only thusly registered shapes can be used by issuing the command `shape(shapename)`.

79.4 Links

- http://openbookproject.net/thinkcs/python/english3e/hello_little_turtles.html
- <https://docs.python.org/3/library/turtle.html>

79.5 Robot Dance Simulator

```
cms robot dance dance.txt
```

79.6 Scratch

- [Scratch](#)

79.7 MBlock

- [MBlock](#)



80. Tools

- **Terminal:** On OSX, when you navigate to the search magnification glass, you can type in *terminal* to start it. A terminal allows you to execute a number of commands to interact with the computer from a commandline interface, e.g. the terminal.
- **Bash** is the command language used in terminal.
- **Pyenv** allows to manage multiple versions of python easily. [Pyenv link](#)
- **XCode** is an integrated development environment for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, watchOS and tvOS.
- **Homebrew** is a *package manager* for OS X which lets the user *install software* from *UNIX* and *open source software* that is not included in OSX.
- **pyCharm:** is an Integrated Development Environment for Python.
- **Matplotlib:** Matplotlib is a library that allows us to create nice graphs in python. As we typically install python with virtualenv, we need to configure matplotlib properly to use it. The easiest way to do this is to execute the following commands. After you run them you can use matplotlib.

```
$ pip install numpy
$ pip install matplotlib
$ echo "backend : TkAgg" > ~/.matplotlib/matplotlibrc
```
- **Macdown** a macdown editor for OSX
- **Markdown** (from Markdown)
- **AquaEmacs** (from Aquaemacs)
- **Marvelmind** (from Marvelmind if you have marvelmind positioning sensors which are optional)
- **Arduino** (from Arduino if you like to use their interface to access the esp8266 boards)
- **40 OSX Terminal Tricks**

80.1 Markdown

Markdown is a format convention that produces nicely formatted text with simple ASCII text. Markdown has very good support for editors that render the final output in a view window next to the editor pane. Two such editors are

- [Macdown](#): MacDown provides a nice integrated editor that works well.
- [pyCharm](#): We have successfully used Vladimir Schneiders [Markdown Navigator plugin](#). Once installed you click on a .md file pycharm will automatically ask to install the plugins from Markdown for you.

A detailed set of syntax rules can be found at: **BUG: LINK TO MARKDOWN SYNTAX MISSING**

The following are some basic examples

- To *emphasize* a text you use `*emphasize*`
- To make text **bold** use `**bold**`
- To make text ***bold-and-emphasize*** use `***bold-and-emphasize***`
- To create a hyperlink use `[Google](https://google.com)` which will result in [Google](https://google.com)
- To include an image use `![Bracketed Text](link)`

A list can be created by item starting with `*`, `a -`, or `a +` or a number

- ```

1. one
2. two

1. one
2. two
 * one
 * two

• one
• two

```

If you need to indent items underneath already bulleted items, precede the indent items with four spaces and they will be nested under the item above them.

To quote text precede it with a `>`.

```

> Quote

Quote

```

Other syntax options can be found in the Format drop-down at the top of the screen between View and Plug-ins of macdown.

## 80.2 Aquamacs

There are many different versions of emacs available on OSX. Aquamacs is often used as it integrates nicely with the OSX GUI interface.

- [AquaEmacs](#)

*Aquamacs* is a program for Mac devices which allows the user to edit text, HTML, LaTeX, C++, Java, Python, R, Perl, Ruby, PHP, and more. Aquaemacs integrates well with OSX and provides many functions through a menu. You will mostly be using the File, Edit, menus or toolbar icons.

Emacs provides convenient keyboard shortcuts, most of which are combinations with the Control

or Meta key (The Meta key is the ESC key). If you accidentally end up doing something wrong simply press CTRL-g to get out without issue. Other Keyboard Shortcuts include:

- CTRL-x u or File>Undo will cancel any command that you did not want done. (CHECK)
- ESC-g will cancel any command you are in the middle of.
- You can break paragraph lines with Ctrl-x w, where w will wrap text around word boundaries.
- To delete text to the end of the current word, press ESC-d.
- to delete the whole line from the position of the cursor to the end, press CTRL-k.

## 80.3 Bash

Bash is preinstalled in OSX. A *bash* script contains *commands* in plain text. In order to create a bash script please decide for a convenient name. Lets assume we name our script *myscript*. Then you can create and edit such a script with

```
$ touch myscript.sh
$ emacs myscripts.sh
```

Next you need to add the following line to the top of the script:

```
#!/bin/bash
```

To demonstrate how to continue writing a script we will be using the bash echo command that allows you to print text. Lets make the second line

```
echo "Hello World"
```

You can now save and start executing your script. Click “File” and then “Save”. Open Terminal and type in cd followed by the name of the folder you put the document in. Now we need to execute the script.

*Executing* a Bash script is rather easy. In order to execute a script, we need to first execute the *permission set*. In order to give Terminal permission to read/execute a Bash script, you have to type

```
chmod u+x myscript.sh
```

After the script has been granted permission to be executed, you can test it by typing

```
./myscript.sh
```

into the terminal. You will see it prints

```
Hello World
```

## 80.4 Arduino

This installation is optional. In the event that there is a TTY error, you will need to install Arduino, since your Mac may be missing some drivers that are included in Arduino. Simply go to [Arduino](#) and follow the installation instructions.

## 80.5 OSX Terminal

[CoolTerm](#)

download <http://freeware.the-meiers.org/CoolTermMac.zip>





# Draft Chapters and Parts





# Draft: Incomming Contributions

|       |                                                     |
|-------|-----------------------------------------------------|
| 80.6  | Python Apache Avro                                  |
| 80.7  | Creating a Virtual Machine on Google Compute Engine |
| 80.8  | PI DHCP Server Tutorial                             |
| 80.9  | VERSION A: Install Docker on a Raspberry Pi         |
| 80.10 | VERSION B: Pi Docker                                |
| 80.11 | Rasberry Pi Hadoop Spark Cluster                    |
| 80.12 | Raspberry Pi Kubernetes Cluster                     |
| 80.13 | Intructions for installing kubernetes               |
| 80.14 | Install Raspbian on an SD card using MacOS          |
| 80.15 | Adding a Public keys to a Raspbery Pi.              |

|           |                                    |            |
|-----------|------------------------------------|------------|
| <b>81</b> | <b>Contributed Tutorials</b> ..... | <b>849</b> |
|-----------|------------------------------------|------------|

|      |            |
|------|------------|
| 81.1 | Open Stack |
| 81.2 | SSH        |
| 81.3 | MapReduce  |
| 81.4 | PI         |
| 81.5 | Aws        |
| 81.6 | Other      |





## 80.6 Python Apache Avro

Min Chen (hid-sp18-405)

Apache Avro is a data serialization system, which provides rich data structures, remote procedure call (RPC), a container file to store persistent data and simple integration with dynamic languages [42]. Avro depends on schemas, which are defined with JSON. This facilitates implementation in other languages that have the JSON libraries. The key advantages of Avro are schema evolution - Avro will handle the missing/extra/modified fields, dynamic typing - serialization and deserialization without code generation, untagged data - data encoding and faster data processing by allowing data to be written without overhead.

The following steps illustrate using Avro to serialize and deserialize data with example modified from Apache Avro 1.8.2 Getting Started (Python) [44].

### 80.6.1 Download, Unzip and Install

The zipped installation file *avro-1.8.2.tar.gz* could be downloaded from [here](#)

To unzip, using linux:

```
1 tar xvf avro-1.8.2.tar.gz
```

using MacOS:

```
1 gunzip -c avro-1.8.2.tar.gz | tar xopf -
```

cd into the directory and install using the following:

```
1 cd avro-1.8.2
2 python setup.py install
```

To check successful installation, import avro in python without error message:

```
1 python
2 >>> import avro
```

This above instruction is for Python2. The Python3 counterpart, *avro-python3-1.8.2.tar.gz* could be downloaded from [here](#) and the unzip and install procedure is the same.

### 80.6.2 Defining a schema

Use a simple schema for students contributed in cloudmesh as an example: paste the following lines into an empty text file with the name it *student.avsc*

```
1 {"namespace": "cloudmesh.avro",
2 "type": "record",
3 "name": "Student",
4 "fields": [
5 {"name": "name", "type": "string"},
6 {"name": "hid", "type": "string"},
7 {"name": "age", "type": ["int", "null"]},
8 {"name": "project_name", "type": ["string", "null"]}
```

```

9]
10 }

```

This schema defines a record representing a hypothetical student, which is defined to be a record with the name *Student* and 4 fields, namely name, hid, age and project name. The type of each of the field needs to be provided. If any field is optional, one could use the list including *null* to define the type as shown in age and project name in the example schema. Further, a namespace *cloudmesh.avro* is also defined, which together with the name attribute defines the full name of the schema (cloudmesh.avro.Student in this case).

### 80.6.3 Serializing

The following piece of python code illustrates serialization of some data

```

1 import avro.schema
2 from avro.datafile import DataFileWriter
3 from avro.io import DatumWriter
4
5 schema = avro.schema.parse(open("student.avsc", "rb").read())
6
7 writer = DataFileWriter(open("students.avro", "wb"), DatumWriter(), schema)
8 writer.append({"name": "Min Chen", "hid": "hid-sp18-405", "age": 29,
9 "project_name": "hadoop with docker"})
10 writer.append({"name": "Ben Smith", "hid": "hid-sp18-309",
11 "project_name": "spark with docker"})
12 writer.append({"name": "Alice Johnson", "hid": "hid-sp18-208", "age": 27})
13 writer.close()

```

The code does the following:

- Imports required modules
- Reads the schema *student.avsc* (make sure that the schema file is placed in the same directory as the python code)
- Create a *DataFileWriter* called *writer*, for writing serialized items to a data file on disk
- Use *DataFileWriter.append()* to add data points to the data file. Avro records are represented as Python dicts.
- The resulting data file saved on the disk is named *students.avro*
- This above instruction is for Python2. If one is using Python3, change

```

1 schema = avro.schema.parse(open("student.avsc", "rb").read())

```

to:

```

1 schema = avro.schema.Parse(open("student.avsc", "rb").read())

```

since the method name has a different case in Python3.

### 80.6.4 Deserializing

The following python code illustrates deserialization

```

1 from avro.datafile import DataFileReader
2 from avro.io import DatumReader
3
4 reader = DataFileReader(open("students.avro", "rb"), DatumReader())
5 for student in reader:

```

```
6 print (student)
7 reader.close()
```

The code does the following:

- Imports required modules
- Use *DatafileReader* to read the serialized data file *students.avro*, it is an iterator
- Returns the data in a python dict

The output should look like:

```
1 {'name': 'Min Chen', 'hid': 'hid-sp18-405',
2 'age': 29, 'project_name': 'hadoop with docker'}
3 {'name': 'Ben Smith', 'hid': 'hid-sp18-309',
4 'age': None, 'project_name': 'spark with docker'}
5 {'name': 'Alice Johnson', 'hid': 'hid-sp18-208',
6 'age': 27, 'project_name': None}
```

### 80.6.5 Resources

- The steps and instructions are modified from [Apache Avro 1.8.2 Getting Started \(Python\)](#) [44].
- The Avro Python library does not support code generation, while Avro used with Java supports code generation, see [Apache Avro 1.8.2 Getting Started \(Java\)](#) [43].
- Avro provides a convenient way to represent complex data structures within a Hadoop MapReduce job. Details about Avro are documented in [Apache Avro 1.8.2 Hadoop MapReduce guide](#) [45].
- For more information on schema files and how to specify name and type of a record can be found at [record specification](#) [46].

## 80.7 Creating a Virtual Machine on Google Compute Engine

- Bertolt Sobolik (hid-sp18-419) ## Introduction

Google Compute Engine is an Infrastructure as a Service (IaaS) offering from Google. The company offers a wide variety of virtual machines, operating systems, and storage options to serve a wide variety of customers, including many high-end options and configurations designed for specific applications.

### 80.7.1 Requirements

In order to create a VM on Google Compute Engine one must have a Google Account. You can sign up for one [here](#).

Next, you will need to [sign up for Google Cloud Platform](#). As of this writing, Google is offering a free trial that includes \$300 worth of credits that expire one year from signup, but you will need to provide billing information (either a credit card or bank details) to sign up~[[hid-sp18-419-tutorial-gce-signup](#)].

When you sign up, Google will create a project for you called *My First Project* with an autogenerated project ID. The page after the welcome screen contains links to a number of easy to follow tutorials and *Quick Starts* that will guide you through the steps required to start various Google Cloud Platform services in the console.

## 80.7.2 Automated Creation of a VM on Ubuntu 16.04

The following instructions are adapted from [here](#)~[[hid-sp18-419-tutorial-gce-setup](#)].

You will need to have *curl* installed. You can install it with:

```
1 sudo apt install curl
```

First you need to install the gcloud SDK:

```
1 export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
2 echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" \
3 | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
4 curl https://packages.cloud.google.com/apt/doc/apt-key.gpg \
5 | sudo apt-key add -
6 sudo apt-get update && sudo apt-get install google-cloud-sdk
```

Next, initialize the SDK with:

```
1 gcloud init
```

This will open a web browser and prompt you to log in to your account. If you don't have a browser installed or are doing a headless setup, you can use the following command instead:

```
1 gcloud init --console-only
```

This will display a link in your terminal that you must copy and paste into a browser. The browser will return a verification code for you to type into the terminal.

After you have logged in, the terminal will prompt you to select a project or create a new one. Project IDs must be unique. If you pick one like *test-vm* it will fail. If you start again with:

```
1 gcloud init
```

You will be prompted to pick up where you left off or create a new configuration.

After selecting a project, you will be asked if you want to configure a default Compute Region and Zone. If you do not, configuration is complete.

## 80.7.3 Create an Instance

You can create an instance with the following command (where is the name of the instance you want to create):

```
1 gcloud compute instances create <name of instance>
```

If you have not set up a default Compute Region and Zone, you will be prompted to select one from the 45 possibly zones, so it is probably better to either set a default or decide which zone you want the instance in before you type the command. For example, if you want to create an instance called *foo* in *us-central1-a* (which is in Iowa), you would enter:

```
1 gcloud compute instances create foo --zone=us-central-1a
```

This will create an *n1-standard-1* instance with one CPU and 3.75 GB of memory, which costs about \$25 a month. If you leave it running, you will burn through a lot of your free credits for nothing. You can stop it with:

```
1 gcloud compute instances stop foo --zone=us-central-1a
```

A full list of all the options for the `gcloud compute instances` command is [here](#)~[hid-sp18-419-tutorial-gce-reference

## 80.7.4 MongoDB Tutorial

**THIS IS A DRAFT AND NOT COMPLETED YET**

- status: 0
- feedback: 1

Some steps are copied from [https://www.tutorialspoint.com/mongodb/mongodb\\_environment.htm/](https://www.tutorialspoint.com/mongodb/mongodb_environment.htm/)

### Overview

| RDBMS               | MongoDB            |
|---------------------|--------------------|
| Database            | Database           |
| Table               | Collection         |
| Tuple/Row<br>column | Document<br>Field  |
| Table Join          | Embedded Documents |
| Primary Key         | Primary Key        |

MongoDB is an open-source document database and NoSQL database, which is written in C++. A document is a set of key-value pairs. The table shows the difference between RDBMS terminology and MongoDB.

### Advantages of MongoDB over RDBMS

- Schema less - MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- Ease of scale-out - MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

### Install MongoDB on Ubuntu

Run the following command to import the MongoDB public GPG key

```
1 sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

Create a `/etc/apt/sources.list.d/mongodb.list` file using the following command.

```
1 echo 'deb http://downloads-distrow.mongodb.org/repo/ubuntu-upstart dist 10↔
↔ gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
```

Now issue the following command to update the repository

```
1 sudo apt-get update
```

Next install the MongoDB by using the following command

```
1 apt-get install mongodb-10gen = 2.2.3
```

Start MongoDB

```
1 sudo service mongodb start
```

Stop MongoDB

```
1 sudo service mongodb stop
```

Restart MongoDB

```
1 sudo service mongodb restart
```

## 80.8 PI DHCP Server Tutorial

TODO: Min, Bertholt: could som of the actions not be done when we burn the SD Cards. E.g. I wonder if i plugin an SDcard can we not set up all the file actions. Also even whne we are locally on the machine, as this is the first install, could we not just copy or cat the information into the right files. Maybe we can werie a cmd5 command cms pi dhcp ... to make this easy?

- Min Chen (hid-sp18-405)
- Bertolt Sobolik (hid-sp18-419)

We describe how to set up **D**ynamic **H**ost **C**onfiguration **P**rotocol (DHCP) server on a Raspberry Pi Cluster. The OS on these Pi's is RASPBIAN STRETCH WITH DESKTOP released on 2017-11-29.

### 80.8.1 Introduction

“The Dynamic Host Configuration Protocol (DHCP) enables any of the computers on the local area network (LAN) to be given a network configuration automatically as soon as the boot process on the machine gets underway” [595]. Instead of using a router, we use one of the raspbery PI's to fulfill this function. The other Pi's are configured in such a way that the serve as clients and obtain the network address form our raspbery providing the addresses. They are DHCP Clients, and to use DHCP they need to have their networking setup properly configured.

Hence, we set up one of the five Pi's in the cluster (hostname: red00) to be the DHCP server, and the rest four with Pi's with the names red01, red02, red03, red04 will then be DHCP clients.

The information which is passed from DHCP Server to its clients includes[595]:

- a suitable IP Address;
- the address of your router;
- an address of one or more Domain Name Servers (DNS)

If you want to get an introduction about the logical process followed by a DHCP service, please follow [this link](#) [532]

## 80.8.2 Setting up the DHCP server

Choose one of the Pi's as the DHCP server, using the Pi with hostname *red00* as an example here. Log into this Pi and open a terminal. The following steps are all processed in the terminal of this chosen Pi.

### Software installation

The first step is to install a package `dhcpd`, which is a popular DHCP server for the Pi. In the terminal

```
1 sudo apt-get update
2 sudo apt-get install isc-dhcp-server
```

At the end of the installation process, the DHCP server daemon will be started and it will **fail**, because the configuration has not been done. It will get fixed in later steps.

### Configure the DHCP server

The configuration file for the DHCP server is at `/etc/dhcp/dhcpd.conf`. Start the editing process with `nano` as follows:

```
1 sudo nano /etc/dhcp/dhcpd.conf
```

Define subnet which will be the network that all the other Pi's will connect to. Add the following lines to the file `/etc/dhcp/dhcpd.conf`:

```
1 subnet 192.168.2.0 netmask 255.255.255.0 {
2 range 192.168.2.100 192.168.2.200;
3 option broadcast-address 192.168.2.255;
4 option routers 192.168.2.1;
5 max-lease-time 7200;
6 option domain-name "red00";
7 option domain-name-servers 8.8.8.8;
8 }
```

Note: The subnet and netmask are IP values required for assisting communications across your LAN.

- `subnet`: “you can obtain the IP Address of a computer on your LAN using the Linux `ifconfig` command: take the *Inet Addr* value and replace its final octet with a zero to get your subnet” [595];
- `range`: this is the range of IP Addresses distributed by this DHCP Service. You may have two ranges such as

```
1 range 192.168.2.100 192.168.2.120
2 range 192.168.2.150 192.168.2.200
```

to refrain the DHCP server from handling out some of the addresses (from 121 to 149)

- `domain-name-server`: If you have a DNS service for machines on your LAN, enter the server IP address or you can use public DNS Services such as Google's, which are at 8.8.8.8. and 8.8.4.4.

Finally, Save your changes to the file with `Ctrl-0` and exit nano with `Ctrl-X`.



## Change the interface of DHCP service

Now you need to tell the DHCP service the interface to hand out addresses on. Edit the following file:

```
1 sudo nano /etc/default/isc-dhcp-server
```

Find the following section:

```
1 # On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
2 # Separate multiple interfaces with spaces, e.g. "eth0 eth1".
3 INTERFACES=""
```

And change the last line to:

```
1 INTERFACES="eth0"
```

## Set static IP Address for the server

The next step is to set a static IP address on the Raspberry pi as this won't be able to start the DHCP service without it. We use *nano* to edit the file at */etc/network/interfaces*:

```
1 sudo nano /etc/network/interfaces
```

Add the following lines:

```
1 auto eth0
2 iface eth0 inet static
3 address 192.168.2.1
4 netmask 255.255.255.0
```

If you have a line like:

```
1 iface eth0 inet dhcp
```

be sure to disable this line. It is used to set up the Pi as a DHCP client, however, we want this Pi to be a DHCP server.

Now this Raspberry Pi will now always have the IP address 192.168.2.1. You can double-check this by entering the command `ifconfig`; the IP address should be shown on the second line just after `inet addr`.

## Restart the DHCP service

Finally, to complete the set-up, restart the DHCP service by the following command:

```
1 sudo service isc-dhcp-server stop
2 sudo service isc-dhcp-server start
```

## Checking the currently leased addresses

Run the following command to check the currently assigned addresses:

```
1 cat /var/lib/dhcp/dhcpd.leases
```

You should expect something like the following:

```
1 lease 192.168.2.102 {
2 starts 0 2018/02/25 21:36:16;
3 ends 0 2018/02/25 21:46:16;
4 tstp 0 2018/02/25 21:46:16;
5 cltt 0 2018/02/25 21:36:16;
6 binding state active;
7 next binding state free;
8 rewind binding state free;
9 hardware ethernet b8:27:eb:42:c9:e9;
10 uid "\001\270'\353B\311\351";
11 set vendor-class-identifier = "dhcpcd-6.11.5:Linux-4.9.59-v7+:armv7l:↔
↔ BCM2835";
12 client-hostname "red01";
13 }
14 lease 192.168.2.100 {
15 starts 0 2018/02/25 21:41:09;
16 ends 0 2018/02/25 21:51:09;
17 tstp 0 2018/02/25 21:51:09;
18 cltt 0 2018/02/25 21:41:09;
19 binding state active;
20 next binding state free;
21 rewind binding state free;
22 hardware ethernet b8:27:eb:60:b8:8e;
23 uid "\001\270'\353'\270\216";
24 set vendor-class-identifier = "dhcpcd-6.11.5:Linux-4.9.59-v7+:armv7l:↔
↔ BCM2835";
25 client-hostname "red03";
26 }
27 lease 192.168.2.101 {
28 starts 0 2018/02/25 21:41:10;
29 ends 0 2018/02/25 21:51:10;
30 tstp 0 2018/02/25 21:51:10;
31 cltt 0 2018/02/25 21:41:10;
32 binding state active;
33 next binding state free;
34 rewind binding state free;
35 hardware ethernet b8:27:eb:9a:55:13;
36 uid "\001\270'\353\232U\023";
37 set vendor-class-identifier = "dhcpcd-6.11.5:Linux-4.9.59-v7+:armv7l:↔
↔ BCM2835";
38 client-hostname "red02";
39 }
40 lease 192.168.2.103 {
41 starts 0 2018/02/25 21:41:11;
42 ends 0 2018/02/25 21:51:11;
43 tstp 0 2018/02/25 21:51:11;
44 cltt 0 2018/02/25 21:41:11;
45 binding state active;
46 next binding state free;
47 rewind binding state free;
48 hardware ethernet b8:27:eb:3a:7c:7c;
49 uid "\001\270'\353:|";
50 set vendor-class-identifier = "dhcpcd-6.11.5:Linux-4.9.59-v7+:armv7l:↔
↔ BCM2835";
51 client-hostname "red04";
52 }
```

Note that in this case, the 4 Pi's with hostnames: red01, red02, red03, and red04 are assigned IP addresses 192.168.2.102, 192.168.2.101, 192.168.2.100 and 192.168.2.103 respectively

An alternative way is to use the following command:

```
1 dhcp-lease-list --lease PATH_TO_LEASE_FILE
```

which would give a cleaner look such as:

```
1 Reading leases from /var/lib/dhcp/dhcpd.leases
2 MAC IP hostname valid until manufacturer
3 =====
4 b8:27:eb:3a:7c:7c 192.168.2.103 red04 2018-02-25 21:56:13 -NA-
5 b8:27:eb:42:c9:e9 192.168.2.102 red01 2018-02-25 21:50:30 -NA-
6 b8:27:eb:60:b8:8e 192.168.2.100 red03 2018-02-25 21:56:11 -NA-
7 b8:27:eb:9a:55:13 192.168.2.101 red02 2018-02-25 21:56:12 -NA-
```

### 80.8.3 Configure fixed IP's for clients

It is sometimes needed to have the dhcp server assign fixed addresses to each node in the cluster so that it is easy to remember the node by IP addresses. For instance next node in the cluster is red01 and it would be helpful to have a fixed IP for example 192.168.2.50.

To do this, we modify the `/etc/dhcp/dhcpd.conf` by:

```
1 sudo nano /etc/dhcp/dhcpd.conf
```

and add the following lines:

```
1 host red01 {
2 hardware ethernet b8:27:eb:42:c9:e9;
3 fixed-address 192.168.2.50;
4 }
```

Notice that `b8:27:eb:42:c9:e9` is the so-called MAC Address of the Ethernet interface (network adapter) of the machine which you wish to name red01. "It provides a hardware reference on the client for the server to use in network communications. You can find the MAC Address(es) of the Ethernet interface(s) on any computer using the `ifconfig` command" [595]. You may also notice that the previous two commands of checking the currently leased addresses will also provide MAC Addresses.

**Warning:** "if you wish to have your DHCP Server award a fixed IP Address it should be one outside the DHCP normally assigned range of IP Addresses" [595].

Another thing to notice is that the previous two commands of checking the currently leased addresses does not include the clients given a fixed address. For example, if one runs the command after the fixed IP config for red01:

```
1 dhcp-lease-list --lease PATH_TO_LEASE_FILE
```

the result would be:

```
1 Reading leases from /var/lib/dhcp/dhcpd.leases
2 MAC IP hostname valid until manufacturer
3 =====
4 b8:27:eb:3a:7c:7c 192.168.2.103 red04 2018-02-25 21:56:13 -NA-
5 b8:27:eb:60:b8:8e 192.168.2.100 red03 2018-02-25 21:56:11 -NA-
6 b8:27:eb:9a:55:13 192.168.2.101 red02 2018-02-25 21:56:12 -NA-
```

red01 is no longer in the list because it is assigned a fixed-address 192.168.2.50

### Checking the currently leased addresses for fixed IP clients

To check the currently leased addresses for fixed IP clients:

```
1 cat /var/lib/dhcp/dhclient.eth0.leases
```

## 80.8.4 Resources

The steps and instruction presented here are combined from several web resources:

- [Lesson 3 - Dynamic Host Configuration Protocol \(DHCP\) \[532\]](#)
- [Configuring the Raspberry Pi as a DHCP Server under Raspbian Wheezy \[595\]](#)
- [Hadoop and Spark Installation on Raspberry Pi-3 Cluster \[615\]](#)
- [Setup Raspberry pi as a dhcp server \[284\]](#)

## 80.9 VERSION A: Install Docker on a Raspberry Pi

hid-sp18-503

Docker is now supported on ARM processors. Thus the installation of docker on Raspberry PI's is extremely simple and we do not need any special setup to get docker running.

To install docker on the raspberry pi, please execute the following steps:

First, ssh into the raspberry pi (or login using a monitor and open terminal) Second, run the following command

```
curl -sSL https://get.docker.com | sh
```

If you dont want to run docker using sudo add pi to the docker user group as recommended using

```
sudo usermod -aG docker pi
```

Now any docker image built for ARM can be run. Naturally it must be small enough to fit on the PI. Please remember it has a very small memory.

Please report back to us if you found useful packages and tools for the PI.

## 80.10 VERSION B: Pi Docker

Docker is a tool that allows you to deploy applications inside of software containers. It is a method of packaging software, to include not only your code, but also other components such as a full file system, system tools, services, and libraries. This can be useful for the Raspberry Pi because it allows users to run applications without lot of steps, as long as the application is packaged inside of a Docker image. You simply install Docker and run the container.

### 80.10.1 Preparing the SD card

Download the latest Raspbian Jessie Lite image from

```
1 https://www.raspberrypi.org/downloads/raspbian/
```

Please note that Raspbian Jessie Lite image contains the only the bare minimum amount of packages.

### 80.10.2 Download Etcher here:

```
1 https://etcher.io/
```

Now follow the instructions in Etcher to flash Raspbian image on the SD card. Plbefore ejecting the SD card.

### 80.10.3 Enable SSH on the SD Card

To prevent Raspberry Pis from being hacked the RPi foundation have now disabled SSH on the default image. So, create a text file in /boot/ called ssh - it can be empty file or you can type anything you want inside it.

Please note that you have renamed the ssh.txt to ssh i.e. without extension.

Now insert the SD card, networking and power etc.

### 80.10.4 Starting Pi

Once you boot up the Raspberry Pi, Connect using SSH

```
1 $ ssh pi@raspberrypi.local
```

The password is raspberry.

For security reasons, please change the default password of the user pi using the passwd command.

Note : If you want to change the hostname of the Pi, Use an editor and change the hostname raspberrypi in:

```
1 * /etc/hosts
2 * /etc/hostname
```

### 80.10.5 Docker Installation

### 80.10.6 Run apt-get update

Since Raspbian is Debian based, we will use apt to install Docker. But first, we need to update.

```
1 sudo apt-get update
```

### 80.10.7 Install Docker

An automated script maintained by the Docker project will create a systemd service file and copy the relevant Docker binaries into /usr/bin/.

```
1 $ curl -sSL https://get.docker.com | sh
```

### 80.10.8 Configure Docker

```
1 * Set Docker to auto-start
2
3 $ sudo systemctl enable docker
4
5 * Reboot the Pi, or start the Docker daemon with:
6
7 $ sudo systemctl start docker
```

### 80.10.9 Enable Docker client

The Docker client can only be used by root or members of the docker group. Add pi or your equivalent user to the docker group using :

```
1 $ sudo usermod -aG docker pi
```

After executing the above command, log out and reconnect with ssh.

### 80.10.10 Test Docker

To test docker was installed successfully, run the hello-world image.

```
1 $ docker run hello-world
```

If Docker is installed properly, you'll see a "Hello from Docker!" message.

## 80.11 Raspberry Pi Hadoop Spark Cluster

hid-sp18-412 hid-sp18-410 hid-sp18-408 hid-sp18-406

### 80.11.1 Initial Configuration

a> Login to the terminal and change password for the Raspberry pi

```
1 passwd pi
2 Enter the password as the snowcluster
```

b> Next, use raspi-config to configure the OS from a terminal window

```
1 sudo raspi-config in a terminal window
```

c> In the configuration window do the following

```
1 1. Select Interfacing Options
2 2. Navigate to and select SSH
3 3. Choose Yes
4 4. Select Ok
5 5. Choose Finish
```

d> Now, we need to check if we are able to connect to the raspberry pi from the different pc or another raspberry pi. Login to the Raspberry pi 1, and type in a terminal the below command

```
1 ifconfig
```

We use this command to identify the etho ipaddress returned via `ifconfig` command. We connect the same ethernet to a laptop and do the ssh to the respective raspberry pi 1 ip address to check if the ssh has been successfully enabled.

e> Next repeat the steps from the a to d on all other 4 Rasberry pi's.

```
1 HostNames
2 sudo vi /etc/hosts
3
4 piHadoopmaster 169.254.24.132
5 piHadoopslave1 169.254.35.145
6 piHadoopslave2 169.254.87.91
7 piHadoopslave3 169.254.225.63
8 piHadoopslave4 169.254.190.73
```

f> After changing then reboot all the nodes.

g> Login to the worker1 (piHadoopslave1) then, open the `etc/hosts` and update it as below

```
1 169.254.35.145 PiHadoopSlave1
2 169.254.24.132 PiHadoopMaster
```

Then reboot and verify the hostname with command:

```
1 hostname
2 hostname -i
```

h> Now repeat this step g for all other 3 workers.

### 80.11.2 Creating a new group and user

Perform this step on master and other 4 workers

a> Connect to the master (make sure to check the hostname)

b> Create hadoop group with the below command

```
sudo addgroup hadoop
```

c> Add the hduser to the hadoop group which we have created.

```
sudo adduser --ingroup hadoop hduser
```

d> Add hduser to the sudoers list

```
sudo adduser hduser sudo
```

e> Switch user to the hduser

```
su hduser
```

(password: snowcluster)

### 80.11.3 Generating the ssh keys (Perform these steps in master and all the workers)

a> Generate the ssh key using the following commands.

```
1 cd ~
2 mkdir .ssh
3
4 ssh-keygen -t rsa -P ""
```



```

5
6 cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
7 chmod 600 authorized_keys

```

b> Copy this key from master to other workers to enable the password less ssh using following command

```
1 ssh-copy-id -i ~/.ssh/id_rsa.pub <hostname of master/slave-1>
```

c> Make sure you are able to verify password less ssh using following command

```
1 $ ssh <hostname of master/slave-1>
```

d> Login to the hduser

```
1 su hduser
```

e> Copy the contents from the id\_rs.pub to the authorized\_keys

```
1 sudo cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
```

#### 80.11.4 Installing Hadoop

a> Download hadoop

```
1 wget ftp://apache.belnet.be/mirrors/ftp.apache.org/hadoop/common/hadoop↔
↔ -2.7.1/hadoop-2.7.1.tar.gz
```

b> Create a new directory opt (if the directory does not exist)

```
1 sudo mkdir /opt
```

c> Navigate to the home directory cd ~ d> Unzip and change ownership sudo tar -xvzf hadoop-2.7.1.tar.gz -C /opt/ cd /opt sudo chown -R hduser:hadoop hadoop-2.7.1/

#### 80.11.5 Setting the Environment Variables by modifying the bashrc file

Perform all these steps in master and all other workers

a> Open the bashrc file vi ~/.bashrc

b> Copy the following lines at the end of the bashrc file

```

1 export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:jre/bin/java::")
2 export HADOOP_HOME=/opt/hadoop-2.7.1
3 export HADOOP_MAPRED_HOME=$HADOOP_HOME
4 export HADOOP_COMMON_HOME=$HADOOP_HOME
5 export HADOOP_HDFS_HOME=$HADOOP_HOME
6 export YARN_HOME=$HADOOP_HOME
7 export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
8 export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
9 export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

```

c> Execute bashrc

```

1 source ~/.bashrc
2 hadoop version

```

You will see output similar to

```

1 Hadoop 2.7.1
2 Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 15↵
 ↪ ecc87ccf4a0228f35af08fc56de536e6ce657a
3 Compiled by jenkins on 2015-06-29T06:04Z
4 Compiled with protoc 2.5.0
5 From source with checksum fc0a1a23fc1868e4d5ee7fa2b28a58a
6 This command was run using /opt/hadoop-2.7.1/share/hadoop/common/hadoop-↵
 ↪ common-2.7.1.jar

```

### 80.11.6 Configure hadoop 2.7.1 (Perform this on the master and all the other workers)

a> Login to the master and go to the directory that contains all the configuration files of Hadoop. We want to edit the `hadoop-env.sh` file and we need to configure `JAVA_HOME` manually in this file, Hadoop seems to ignore our existing `JAVA_HOME`. Execute the following command to navigate to the hadoop configuration directory

```
1 cd $HADOOP_CONF_DIR
```

b> Open the `hadoop-env.sh` file.

```
1 vi hadoop-env.sh
```

c> Update export statement

```
1 export JAVA_HOME=/usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/
```

### 80.11.7 Edit the XMI Files

### 80.11.8 Steps for Master

We need to edit `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml` and `yarn-site.xml` files to update the configurations.

a> Navigate to the hadoop configuration directory

```
1 vi core-site.xml
```

b> Then copy paste the following code between the configuration tags to the `core-site.xml`

vi `core-site.xml`

```

1 <configuration>
2 <property>
3 <name>fs.default.name</name>
4 <value>hdfs://PiHadoopMaster:54310</value>
5 </property>
6 <property>
7 <name>hadoop.tmp.dir</name>
8 <value>/hdfs/tmp</value>
9 </property>

```

c> Edit the xml file `hdfs-site.xml`

```
1 vi hdfs-site.xml
```

d> Copy paste the code between the configuration tags appropriately to the `hdfs-site.xml` file  
`dfs.replication 1 fs.default.name hdfs://PiHadoopMaster:54310 hadoop.tmp.dir /hdfs/tmp`

e> Rename the existing `mapred-site.xml.template` to the `mapred-site.xml`

```
1 cp mapred-site.xml.template mapred-site.xml
```

f> Open the `mapred-site.xml` file

```
1 vi mapred-site.xml
```

Then copy the following code to the file

```
1 <property>
2 <name>mapreduce.framework.name</name>
3 <value>yarn</value>
4 </property>
5 <property>
6 <name>mapreduce.map.memory.mb</name>
7 <value>256</value>
8 </property>
9 <property>
10 <name>mapreduce.map.java.opts</name>
11 <value>-Xmx210m</value>
12 </property>
13 <property>
14 <name>mapreduce.reduce.memory.mb</name>
15 <value>256</value>
16 </property>
17 <property>
18 <name>mapreduce.reduce.java.opts</name>
19 <value>-Xmx210m</value>
20 </property>
21 <property>
22 <name>yarn.app.mapreduce.am.resource.mb</name>
23 <value>256</value>
24 </property>
```

Note: The first property tells us that we want to use Yarn as the MapReduce framework. The other properties are some specific settings for our Raspberry Pi. For example we tell that the Yarn Mapreduce Application Manager gets 256 megabytes of RAM and so does the Map and Reduce containers. These values allow us to actually run stuff, the default size is 1.5 GB which our Pi can't deliver with its 1GB RAM.

g> Open the `yarn-site.xml`

```
1 vi yarn-site.xml
```

f> Then copy the following code to the 'yarn-site.xml' file

```
1 <configuration>
2 <property>
3 <name>dfs.replication</name>
4 <value>1</value>
5 </property>
6 <property>
7 <name>fs.default.name</name>
8 <value>hdfs://PiHadoopMaster:54310</value>
9 </property>
```

```

 9 <property>
10 <name>hadoop.tmp.dir</name>
11 <value>/hdfs/tmp</value>
12 </property>
13 </property>
14 <property>
15 <name>yarn.nodemanager.aux-services</name>
16 <value>mapreduce_shuffle</value>
17 </property>
18 <property>
19 <name>yarn.nodemanager.resource.cpu-vcores</name>
20 <value>4</value>
21 </property>
22 <property>
23 <name>yarn.nodemanager.resource.memory-mb</name>
24 <value>1024</value>
25 </property>
26 <property>
27 <name>yarn.scheduler.minimum-allocation-mb</name>
28 <value>128</value>
29 </property>
30 <property>
31 <name>yarn.scheduler.maximum-allocation-mb</name>
32 <value>1024</value>
33 </property>
34 <property>
35 <name>yarn.scheduler.minimum-allocation-vcores</name>
36 <value>1</value>
37 </property>
38 <property>
39 <name>yarn.scheduler.maximum-allocation-vcores</name>
40 <value>4</value>
41 </property>
42 <property>
43 <name>yarn.nodemanager.vmem-check-enabled</name>
44 <value>false</value>
45 <description>Whether virtual memory limits will be enforced for ↵
 ↵ containers</description>
46 </property>
47 <property>
48 <name>yarn.nodemanager.vmem-pmem-ratio</name>
49 <value>4</value>
50 <description>Ratio between virtual memory to physical memory when setting ↵
 ↵ memory limits for containers</description>
51 </property>
52 <property>
53 <name>yarn.resourcemanager.resource-tracker.address</name>
54 <value>RaspberryPiHadoopMaster:8025</value>
55 </property>
56 <property>
57 <name>yarn.resourcemanager.scheduler.address</name>
58 <value>RaspberryPiHadoopMaster:8030</value>
59 </property>
60 <property>
61 <name>yarn.resourcemanager.address</name>
62 <value>RaspberryPiHadoopMaster:8040</value>

```

```
63 </property>
64 </configuration>
```

This file tells Hadoop some information about this node, like the maximum number of memory and cores that can be used. We limit the usable RAM to 768 megabytes, that leaves a bit of memory for the OS and Hadoop. A container will always receive a memory amount that is a multitude of the minimum allocation, 128 megabytes. For example a container that needs 450 megabytes, will get 512 megabytes assigned.

### 80.11.9 Edit XML files for the workers

#### 80.11.10 Prepare the HDFS directory by executing the below commands

```
1 sudo mkdir -p /hdfs/tmp
2 sudo chown hduser:hadoop /hdfs/tmp
3 chmod 750 /hdfs/tmp
4 hdfs namenode -format
```

#### 80.11.11 Booting Hadoop

```
1 cd $HADOOP_HOME/sbin
2 start-dfs.sh
3 start-yarn.sh
```

If you want to verify that everything is working you can use the `jps` command. In the output of this command you can see that Hadoop components like the NameNode are running. The numbers can be ignored, they are process numbers.

```
1 We can see the below output by executing the jps command
2 1297 NameNode
3 1815 NodeManager
4 1578 SecondaryNameNode
5 1387 DataNode
6 1723 ResourceManager
7 2125 Jps
```

#### 80.11.12 Master - Worker Configuration

The host names of the workers should be added in the `slaves` file in the hadoop configuration directory

a> Navigate to Hadoop conf directory

```
1 cd $HADOOP_CONF_DIR
2 vi slaves
```

b> Then copy paste the following content

```
1 PiHadoopMaster
2 PiHadoopSlave1
3 PiHadoopSlave2
4 PiHadoopSlave3
5 PiHadoopSlave4
```

### 80.11.13 Configuration in Worker servers

configuration of the xml files in the below steps

Navigate to the Hadoop Conf directory with the command

```
1 cd $HADOOP_CONF_DIR
```

vi core-site.xml

```
1 <configuration>
2 <property>
3 <name>fs.default.name</name>
4 <value>hdfs://PiHadoopMaster:54310</value>
5 </property>
6 <property>
7 <name>hadoop.tmp.dir</name>
8 <value>/hdfs/tmp</value>
9 </property>
10 </configuration>
11
12 vi hdfs-site.xml
13
14 <configuration>
15 <property>
16 <name>dfs.replication</name>
17 <value>1</value>
18 </property>
19
20 </configuration>
```

cp mapred-site.xml.template mapred-site.xml vi mapred-site.xml

```
1 <configuration>
2 <property>
3 <name>mapreduce.framework.name</name>
4 <value>yarn</value>
5 </property>
6 <property>
7 <name>mapreduce.map.memory.mb</name>
8 <value>256</value>
9 </property>
10 <property>
11 <name>mapreduce.map.java.opts</name>
12 <value>-Xmx210m</value>
13 </property>
14 <property>
15 <name>mapreduce.reduce.memory.mb</name>
16 <value>256</value>
17 </property>
18 <property>
19 <name>mapreduce.reduce.java.opts</name>
20 <value>-Xmx210m</value>
21 </property>
22 <property>
23 <name>yarn.app.mapreduce.am.resource.mb</name>
24 <value>256</value>
25 </property>
```

```

26
27 </configuration>

```

vi yarn-site.xml

```

1 <configuration>
2 <property>
3 <name>yarn.nodemanager.aux-services</name>
4 <value>mapreduce_shuffle</value>
5 </property>
6 <property>
7 <name>yarn.nodemanager.resource.cpu-vcores</name>
8 <value>4</value>
9 </property>
10 <property>
11 <name>yarn.nodemanager.resource.memory-mb</name>
12 <value>768</value>
13 </property>
14 <property>
15 <name>yarn.scheduler.minimum-allocation-mb</name>
16 <value>128</value>
17 </property>
18 <property>
19 <name>yarn.scheduler.maximum-allocation-mb</name>
20 <value>768</value>
21 </property>
22 <property>
23 <name>yarn.scheduler.minimum-allocation-vcores</name>
24 <value>1</value>
25 </property>
26 <property>
27 <name>yarn.scheduler.maximum-allocation-vcores</name>
28 <value>4</value>
29 </property>
30 </configuration>

```

#### 80.11.14 Prepare the HDFS directory by executing the below commands

```

1 sudo mkdir -p /hdfs/tmp
2 sudo chown hduser:hadoop /hdfs/tmp
3 chmod 750 /hdfs/tmp
4 hdfs namenode -format

```

#### 80.11.15 Booting Hadoop

```

1 cd $HADOOP_HOME/sbin
2 start-dfs.sh
3 start-yarn.sh

```

If you want to verify that everything is working you can use the `jps` command. In the output of this command you can see that Hadoop components like the NameNode are running. The numbers can be ignored, they are process numbers.

```

1 We can see the below output by executing the jps command
2 1297 NameNode
3 1815 NodeManager

```



```

4 1578 SecondaryNameNode
5 1387 DataNode
6 1723 ResourceManager
7 2125 Jps

```

### 80.11.16 REBOOT and start services

- a) Reboot the Workers and Master
- b) Start Master start-dfs.sh start-yarn.sh

### 80.11.17 TESTING THE CONFIGURATION OF MASTER\_SLAVE

- a) Make sure all the all the Worker servers are up and running
- b) Execute the below command in Master
 

```
hdfs dfsadmin -report
```

 (All the Live nodes including Master, total 5 nodes will be listed)

### 80.11.18 ISSUES TO BE ADDRESSED

After restarting the slaves, if the above cpmmand does not result 5 data nodes (master and 4 slaves) then do the following on all slaves

If tmp does not exist

```

1 sudo mkdir -p /hdfs/tmp
2 sudo chown hduser:hadoop /hdfs/tmp
3 chmod -R 755 /hdfs/tmp
4 hdfs namenode -format

```

else

```

1 cd $HADOOP_HOME/sbin
2 stop-all.sh
3 rm -Rf /hdfs/tmp/dfs/*
4 hdfs namenode -format

```

## 80.12 Raspberry Pi Kubernetes Cluster

- Tim Whitson @whitstd (hid-sp18-526)
- Juliano Gianlupi @JulianoGianlupi (hid-sp18-601)

### 80.12.1 Note

This tutorial is for Raspbian Stretch.

### 80.12.2 Hardware

Each cluster consists of:

- 1 head node ([setup](#)) (recommend following the instructions here first)
- 4 compute nodes

### 80.12.3 Configuration

#### Flash Raspbian

1. Download Raspbian image <https://www.raspberrypi.org/downloads/>.
2. Download Etcher <https://etcher.io/>.
3. Using Etcher, flash Raspbian onto SD card.

#### Keyboard Layout

The default keyboard layout may need to be changed to US.

Menu -> Preferences -> Mouse and Keyboard Settings -> Keyboard tab -> Variant -> English (US)

#### Change password

Change password:

```
1 passwd
```

Enter new password via prompts.

#### Change hostnames

Change hostname of each raspberry pi (in descending order).

1. rp0
2. rp1
3. rp2
4. rp3
5. rp4

This can be done on the command line using:

```
1 sudo raspi-config
```

Or on the desktop by going to Menu -> Preferences -> Raspberry Pi Configuration

Or by modifying `/etc/hostname`

#### Configure Head Node

Install Dependencies:

```
1 apt-get update
2 apt-get install -q dnsmasq clusterssh iptables-persistent
```

#### Create Static IP

Copy old config (-n flag prevents overwrite):

```
1 \cp -n /etc/dhcpd.conf /etc/dhcpd.conf.old
```

To update DHCP configuration, add the following to `/etc/dhcpd.conf`:

```
1 interface wlan0
2 metric 200
3
4 interface eth0
```

```

5 metric 300
6 static ip_address=192.168.50.1/24
7 static routers=192.168.50.1
8 static domain_name_servers=192.168.50.1

```

### Configure DHCP Server:

Copy old config (-n flag prevents overwrite):

```

1 \cp -n /etc/dnsmasq.conf /etc/dnsmasq.conf.old

```

To update DNS configuration, add the following to **/etc/dhcpd.conf**

```

1 interface=eth0
2 interface=wlan0
3
4 dhcp-range=eth0, 192.168.50.1, 192.168.50.250, 24h

```

### NAT Forwarding

To Setup NAT Forwarding, uncomment the following line in **/etc/sysctl.conf**:

```

1 net.ipv4.ip_forward=1

```

### IP Tables

Create IP Tables:

```

1 sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
2 sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
3 sudo iptables -A FORWARD -i $INTERNAL -o wlan0 -j ACCEPT
4 sudo iptables -A FORWARD -i $EXTERNAL -o eth0 -j ACCEPT

```

Make rules permanent:

```

1 iptables-save > /etc/iptables/rules.v4

```

### SSH Configuration

**Note: Gregor says this is not best practice**

Generate SSH keys:

```

1 ssh-keygen -t rsa

```

Copy key to each compute node:

```

1 ssh-copy-id <hostname>

```

For hostnames rp1-4 (final node names will be: rp0, rp1, rp2, rp3, rp4).

### Configure Cluster SSH

To update Cluster SSH configuration, add the following to **/etc/clusters**:

```

1 rpcluster rp1 rp2 rp3 rp4

```

Now you can run commands to all clusters by:

```

1 cssh rpcluster

```

## 80.13 Instructions for installing kubernetes

### 80.13.1 First install docker, disable swap, install kubeadm

All the following steps are made automatically by the `docker_kubernetes_install.sh` script.

#### Install docker

In order to install kubernetes you first need to have docker installed. This is very strait foward.

#### Disable swap memory

Docker has an issue (in my opinion severe) in that it is **not compatible with SWAP memory**, therefore it is needed to disable it. This might create some issues, if you encounter them you should try to rebot the cluster again, if that fails change line 16 in the script from

```
1 orig="$(head -n1 /boot/cmdline.txt) cgroup_enable=cpuset cgroup_memory=1"
```

to

```
1 orig="$(head -n1 /boot/cmdline.txt) cgroup_enable=cpuset cgroup_memory=↔
↔ memory".
```

#### Installing kubernetes administrator

Finally, to configure kubernetes you'll need kubeadm. Now the Pi needs to be rebooted.

***All of this will be done by the script, don't worry (maybe worry)***

### 80.13.2 For the nodes

All of the above needs to be done in each node aswell. The script `copy_dk_kub_install_script_to_nodes.sh` should copy the needed script to each of them and run it. It is set up to work with 4 nodes named `rp<number>` with `pi` as the username (the numbers start at 1 because the head node is `rp0`). Changing the number of nodes is trivial, if all of your nodes have the same username it is also trivial.

If your nodes are not configured like that you'll need to change this script or copy `docker_kubernetes_install.sh` to each of the nodes manually.

### 80.13.3 Now some more explanations on the scripts

First docker needs to be installed

```
1 curl -sSL get.docker.com \
2 | sh \
3 && sudo usermod pi -aG docker
```

Now, as docker does not work with SWAP memory it needs to be disabled, this is easy enough

```
1 sudo swapoff -a
2 echo Adding " cgroup_enable=cpuset cgroup_enable=1" to /boot/cmdline.txt
3 sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt
```

If after running this script your raspberry pi starts to not work properly reboot it once again, if that does not solve the issue replace `cgroup_memory=1` with `cgroup_enable=memory`.

```
1 orig="$(head -n1 /boot/cmdline.txt) cgroup_enable=cpuset cgroup_memory=1"
2 echo $orig | sudo tee /boot/cmdline.txt
```

Now kubernetes admin will be installed

```
1 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-↵
 ↵ key add - &&\
2 echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | \
3 sudo tee /etc/apt/sources.list.d/kubernetes.list && \
4 sudo apt-get update -q && \
5 sudo apt-get install -qy kubeadm
6 sudo reboot
```

### On the nodes

Docker and kubernetes need to be installed on the nodes, as well as the SWAP memory needs to be disabled. This is handled by another script that simply copies the installation script to the nodes and runs it. First copy the script

```
1 for number in {1..4}
2 do
3 scp /home/pi/docker_kubernites_install.sh \
4 pi@rp$number:/home/pi/docker_kubernites_install.sh
5 done
6 exit 0
```

Now the installation script will be run on the nodes using cssh

```
1 cssh -a "sh docker_kubernites_install.sh"
```

## 80.13.4 Steps for Setting up the Spark on the Raspberry Pi Cluster

Pre-requisite - Hadoop should be installed

hid-sp18-412 hid-sp18-410 hid-sp18-408 hid-sp18-406

Navigate to the path /home/hduser

```
1 cd /home/hduser
```

Begin the download with the following command

```
1 wget http://apache.claz.org/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz
```

Unzip the tar file of the spark

```
1 tar -xzf spark-2.3.0-bin-hadoop2.7.tgz
```

Move the spark that is extracted to the directory /opt/

```
1 sudo mv spark-2.3.0-bin-hadoop2.7 /opt/
```

Navigate to the directory /opt/spark-2.3.0-bin-hadoop2.7/conf

```
1 cd /opt/spark-2.3.0-bin-hadoop2.7/conf
```

Copy the template from spark-env.sh.template to spark-env.sh

```
1 cp spark-env.sh.template spark-env.sh
```

Open spark-env.sh then set the spark host and the worker memory.

```
1 vi spark-env.sh
2
3 SPARK_MASTER_HOST = 169.254.24.132
4 SPARK_WORKER_MEMORY = 512m
```

## 80.14 Install Raspbian on an SD card using MacOS

- Min Chen (hid-sp18-405)
- Yue Guo (hid-sp18-508)

### 80.14.1 Overview

- This section aims at providing instructions for installing a Raspberry Pi operating system, **Raspbian** on an SD card. This SD card can then be used to boot up a Raspberry Pi with the operating system the card carried.
- Another computer with an SD card reader is needed and we assume the operating system of this computer is **MacOS**.
- The general instruction followed is at [INSTALLING OPERATING SYSTEM IMAGES \[530\]](#).
- Section ~80.14.2 follows [INSTALLING OPERATING SYSTEM IMAGES \[531\]](#).
- Section ~80.14.3 follows [NOOBS \[533\]](#).
- Operating system: MacOS

### 80.14.2 Method1 - Without using NOOBS

~

#### Download

- Download page is [here](#)
- Choose Raspbian instead of NOOBS
- Within raspbian, there are two versions, RASPBIAN STRETCH WITH DESKTOP and RASPBIAN STRETCH LITE. The first one was downloaded (fullversion)

#### Writing an image to the SD card

- Followed the instructions at the beginning, download and install [Etcher](#)
- Connect an SD card reader with the SD card inside
- Open Etcher and select from hard drive the Raspberry Pi .img or .zip file to write to the SD card.
- Select the SD card to write the image to.
- Review selections and click *Flash!* to begin writing data to the SD card.

### 80.14.3 Method2 - Using NOOBS

~

## Download

- download page is [here](#)

## How to install NOOBS on an SD card

Once you've downloaded the NOOBS zip file, you'll need to copy the contents to a formatted SD card on your computer. Here are the detailed steps:

- Format an SD card which is 8GB or larger as FAT.
- Download and extract the files from the NOOBS zip file.
- Copy the extracted files onto the SD card that you just formatted, so that this file is at the root directory of the SD card. Please note that in some cases it may extract the files into a folder; if this is the case, then please copy across the files from inside the folder rather than the folder itself.
- On first boot, the *RECOVERY* FAT partition will be automatically resized to a minimum, and a list of OSes that are available to install will be displayed.

### 80.14.4 Resources

- General instructions: [INSTALLING OPERATING SYSTEM IMAGES \[530\]](#).
- Instructions for using NOOBS are from [NOOBS \[533\]](#).
- Instructions without NOOBS are from [INSTALLING OPERATING SYSTEM IMAGES \[531\]](#).

### 80.14.5 Create a Raspian SD-Card from a Ubuntu Machine

hid-sp18-503

- In the file explorer, right click on the sd card and format the sd card
- Run `df -h` to list all the drives in the computer
- Insert the sd card and run the command again
- Now a new entry will be listed which is the sd card
- The left column of the results from `df -h` command gives the device name of your SD card. It will be listed as something like `/dev/mmcblk0p1` or `/dev/sdX1`, where X is a lower case letter indicating the device. The last part (p1 or 1 respectively) is the partition number.
- Note down the name of the sd card (without the partition)
- Unmount the card so that the card can not be read from or written to
- Run the following command:  
`umount dev/mmcblk0p1`  
Make sure to use correct name for the card
- If your card has multiple partitions unmount all partitions
- Next write the image to the sd card.
- Run the following command:  
`dd bs=4M if=<path to .img> of=/dev/mmcblk0 status=progress conv=fsunc`  
Make sure `if=` contains the path to image and `of=` contains the name of the sd card otherwise you may ruin your hard disk

#### Checking that the image was written properly

- Create an image again from the sd card



- Run the following command:  
`dd bs=4M if=/dev/sdX of=from-sd-card.img`
- Truncate the image to be the the same size as that of the raspbian image  
`truncate --reference <original raspbian image> from-sd-card.img`
- Run diff to see if the two files are same
- Run the following command:  
`diff -s from-sd-card.img <odiginal raspbian image>`
- Diff should say that the two files are same

## 80.15 Adding a Public keys to a Raspberry Pi.

hid-sp18-502

hid-sp18-421

status: 10%

### 80.15.1 SD Card method

In this method we copy the public key on the SD card

TODO: hid-sp18-421, hid-sp18-502, Is it possible to do this while plugging in the SD card and add the key there. The provided solution is not scalable.

### 80.15.2 Network Connected Method

In this method we assume the PI is connected to the network

### 80.15.3 Copying the public key by hand

TODO: What if there is already a public key on th PI? Add additionla instructions in another section.

First generate SSH keys on main computer that will be used to connect to the raspberry Pi cluster.

```
1 ssh-keygen -b 2048 -t rsa
```

Hit enter to save the keys in default folder or enter new folder path. Next, it is importnat that you **Enter passphrase**. Do not provide an empty passphrase. Public and private keys generated will be in ~/.ssh folder

- `id_rsa.pub`: This is your public key and will be transferred to your Raspberry PI.
- `id_rsa`: This is your private key which will remain on your main computer you will be using to connect to your Pi.

TODO: This step is unclear. It is not explained what connect your raspberry pi means

Connect to Raspberry Pi and create .ssh directory. Create authorized\_keys file. Change the permissions of files and directory. Use following commands:

```
1 mkdir .ssh
```

```

2 cd .ssh
3 touch authorized_keys
4 chmod o+rwx ~/.ssh/
5 chmod o+rw ~/.ssh/authorized_keys

```

Go back to your main computer and type the following command to transfer the public RSA key to the Raspberry Pi.

```

1 cat ~/.ssh/id_rsa.pub | ssh -p 22 pi@192.168.1.2 'cat >>.ssh/↵
↵ authorized_keys'

```

#### 80.15.4 Travis CI

hid-sp18-503

status: 70

##### Repository Build status



Travis Continuous Integration (Travis CI) is a continuous integration service that synchronizes with github. It allows developers to automate the process of testing an deploying code. Travis CI allows users to test code for various different environments and allows the use of various languages, which can be easily specified by adding a travis.yml file in the repository.

We demonstrate how to use it in Python. To use Travis CI on github you need to have a github account and a a travis account. Please go to the Web sites and creat your account.

- <travis.com>
- <github.com>

To access it within github, you need to allow travis to synchronize the repositories. To do so please go to the configuration .

Next select the repository that you want to use with Travis CI on the travis website. Now go to github and locally clone the repository.

##### The .travis.yml file

In the github repository you need to place a .travis.yml file. The .travis.yml file describes to travis, which language is used, what environments to test the code on, and to install any dependencies that may be required.

The first line of the yml file specifies the language used

```

1 language: python

```

Next we need to specify the versions of python we need to test the code for. Here we need to specify the versions in the same manner as we use to install the language, as travis creates a virtual

environment for each of these versions.

```
1 python:
2 - 3.6.4
```

Dependencies can be specified in the `yml` file under using the `install:` tag. From our Python section ?? we mentioned that the use of `pip` is recommended for python. Thus let us assume we have the requirements in a `requirements.txt` file within the git repository. Hence we can use it in the `install` tag as follows:

```
1 install:
2 - pip install -r requirements.txt
```

If tests should be performed for different versions of libraries, this can also be specified. Let us give an example for different versions of `django`. Here we define the versions in the `env:` tag. Hence the the `install` is run multiple times but for each environment we set.

```
1 env:
2 - django_version=1.10
3 - django_version=1.11
4 install:
5 - pip install Django==$django_version
```

To specify what commands to execute to run the tests, we use the `script:` tag. If a `make` file is used for this purpose, this can be done as follows however `travis` allows the use of `pytest` for this purpose as well.

```
1 script: make test
```

`Travis` allows various services to be run as required at the time of testing. This can be enabled using the `services` tag in the `travis.yml` file. To start a `mongodb` service for testing your code on `travis` add the following to the `travis.yml` file

```
1 services:
2 - mongodb
```

To show, and keep track of the build status of the repository, `travis` allows users to add build status badges in the markdown file. Once a test is complete, click on the badge on the `travis` page, and select `markdown`. Then copy the `markdown` code and add it to the required `markdown` file. On subsequent pushes to the repository, this status keeps updating.

### An Example

TODO: Arnav: the example is not linked. So we need to make sure that the example is also copied. I am not sure what the best way of doing this is, maybe we need a new example directory under the user `cloudmesh` such as `cloudmesh/travis-example`. In any case link to Repo is missing

To showcase the use of `travis`, the `.travis.yml` file in this repository uses a `make` file (`test_build`) to test the `swagger` code. `Swagger` auto generates basic tests that check for correct response codes as specified in the `swagger` specification `yml` file. The `.travis.yml` file specifies that `mongodb` is needed for testing. The `script` tag in `.travis.yml` file invokes the `make` command which in turn performs the following steps.

- It downloads `swagger` cli
- It uses the `swagger/Makefile` to generate the `swagger` code and install requirements

- It install test requirements provided by swagger
- It uses pytest to perform tests that were generated by swagger
- Finally it cleans the repository.

### Travis and docker

TODO: Help needed: we need an extension to showcase how to use docker

### Class examples

A significant example is located at

- <https://github.com/cloudmesh/book/blob/master/.travis.yml>

Here we have published an image to docker hub and use it to verify if this LaTeX is compiling whenever we check in a change to git. Due to the big image, this check will take significant amount of time.

The Dockerfile to create the image is located at

- <https://github.com/cloudmesh/book/tree/master/examples/docker/tex>



## 81. Contributed Tutorials

### 81.1 Open Stack

- <https://github.com/cloudmesh-community/hid-sp18-516/blob/master/tutorial/openstackcluster.md>

### 81.2 SSH

- <https://github.com/cloudmesh-community/hid-sp18-513/blob/master/tutorial/ssh-add.md>
- <https://github.com/cloudmesh-community/hid-sp18-513/blob/master/tutorial/ssh-config.md>
- <https://github.com/cloudmesh-community/hid-sp18-513/blob/master/tutorial/ssh-port-forwarding.md>

### 81.3 MapReduce

- <https://github.com/cloudmesh-community/hid-sp18-409/blob/master/tutorial/spark-udf.md>

### 81.4 PI

- <https://github.com/cloudmesh-community/hid-sp18-401/blob/master/tutorial/pi-dockerswarm.md>
- <https://github.com/cloudmesh-community/hid-sp18-401/blob/master/tutorial/pi-passwordreset.md>
- <https://github.com/cloudmesh-community/hid-sp18-513/blob/master/tutorial/pi-docker.md>
- <https://github.com/cloudmesh-community/hid-sp18-526/blob/master/tutorial/readme-kube.md>

## 81.5 Aws

- <https://github.com/cloudmesh-community/hid-sp18-511/blob/master/tutorial/aws-lambda.md>
- <https://github.com/cloudmesh-community/hid-sp18-420/blob/master/tutorial/aws-account-creation.md>
- <https://github.com/cloudmesh-community/hid-sp18-420/blob/master/tutorial/aws-boto.md>
- <https://github.com/cloudmesh-community/hid-sp18-420/blob/master/tutorial/aws-gui.md>
- <https://github.com/cloudmesh-community/hid-sp18-517/blob/master/tutorial/AWS-EC2-BOTO.md>
- <https://github.com/cloudmesh-community/hid-sp18-517/blob/master/tutorial/AWS-S3-libcloud.md>
- <https://github.com/cloudmesh-community/hid-sp18-518/blob/master/tutorial/boto-s3.md>
- <https://github.com/cloudmesh-community/hid-sp18-518/blob/master/tutorial/libcloud-ec2.md>
- <https://github.com/cloudmesh-community/hid-sp18-506/blob/master/tutorial/Amazon-Kinesis-Data-Streams.md>
- <https://github.com/cloudmesh-community/hid-sp18-506/blob/master/tutorial/amazon-emr.md>
- <https://github.com/cloudmesh-community/hid-sp18-521/blob/master/tutorial/amazon-emr.md>
- <https://github.com/cloudmesh-community/hid-sp18-514/blob/master/tutorial/amazon-emr.md>
- <https://github.com/cloudmesh-community/hid-sp18-402/blob/master/tutorial/SQS-boto.md>
- <https://github.com/cloudmesh-community/hid-sp18-402/blob/master/tutorial/SQS.md>

## 81.6 Other

- <https://github.com/cloudmesh-community/hid-sp18-416/blob/master/tutorial/celery.md>
- <https://github.com/cloudmesh-community/hid-sp18-505/blob/master/tutorial/graphql.md>
- <https://github.com/cloudmesh-community/hid-sp18-507/blob/master/tutorial/README.md>
- <https://github.com/cloudmesh-community/hid-sp18-402/blob/master/tutorial/ECS.md>
- <https://github.com/cloudmesh-community/hid-sp18-404/blob/master/tutorial/mahout.md>
- <https://github.com/cloudmesh-community/hid-sp18-404/blob/master/tutorial/zookeeper.md>
- <https://github.com/cloudmesh-community/hid-sp18-413/blob/master/tutorial/mongoengine.md>
- <https://github.com/cloudmesh-community/hid-sp18-414/blob/master/tutorial/EthereumFullNode.md>
- <https://github.com/cloudmesh-community/hid-sp18-419/blob/master/tutorial/git-fork-into-org.md>



# Draft: Operating System





## 81.7 Ubuntu on a USB stick for OSX

### 81.7.1 Ubuntu on a USB stick for OSX via Command Line

The easiest way to create an ubuntu distribution that can be booted from an USB stick is done via command line. The original Web page for this method is available at

- <https://help.ubuntu.com/community/How%20to%20install%20Ubuntu%20on%20MacBook%20using%20USB%20Stick>

We have copied some of the information from this Web page but made enhancements to it. Currently all images are copied from that Web page.

Our goal is to create a USB stick that has either Ubuntu 16.04.03 or ubuntu 17.10.1 on it. You will need a USB stick/flash drive. We recommend a 8GB or larger. Please let us know if it works for you on larger than 8GB drives.

First you have to download a distribution of your choice. This can be achieved while visiting the URL

- <https://www.ubuntu.com/download/desktop>

We assume that you downloaded to iso from ubuntu to a folder called *iso*. Next we open a terminal and cd into the folder *iso*. Now we need to convert the is to an image file. This is done as follows and you need to execute the command for the version of ubuntu you like to use.

Your folde will look something like this

```
1 ls -l
2
3 ubuntu-16.04.3-desktop-amd64.iso
4 ubuntu-17.10.1-desktop-amd64.iso
```

For 17.10.1 you will need to generate an image with the following command

```
1 hdiutil convert ubuntu-17.10.1-desktop-amd64.iso -format UDRW -o ubuntu↵
 ↵ -17.10.1-desktop-amd64.img
```

For 16.04.3 you will need to generate an image with the following command

```
1 hdiutil convert ubuntu-16.04.3-desktop-amd64.iso -format UDRW -o ubuntu↵
 ↵ -16.04.3-desktop-amd64.img
```

OSX will append a .dmg behind the name. When considering the OS and you only want to use one, we recommend that you use the latest OS. Please let us know if we need to update the verion numbers. Check with the ubuntu Web page.

At this time do not plug in your usb stick. Just issue the command

```
1 diskutil list
```

Observe the output. Now plug in the USB stick. Wait till the USB stick registers in the Finder. If this does not work find a new USB stick or format it. Execute the command

```
1 diskutil list
```

and observer the output again. Another device will register and you will see something like

```

1 /dev/disk2 (external, physical):
2 #: TYPE NAME SIZE IDENTIFIER
3 0: FDisk_partition_scheme *8.2 GB disk2
4 1: DOS_FAT_32 NO NAME 8.2 GB disk2s1

```

Please note in this example the device path and number is recognized as

```
1 /dev/disk2
```

It also says external, which is a good sign as the USB stick is external. Next, we need to unmount the device with

```
1 diskutil unmountDisk /dev/diskN
```

where you replace the number N with the disk number that you found for the device. In our example it would be 2. If you see the error “Unmount of diskN failed: at least one volume could not be unmounted”, start Disk Utility.app and unmount the volume (don’t eject). If it was successful, you will see

```
1 Unmount of all volumes on disk2 was successful
```

The next step is dangerous and you need to make sure you follow it. So please do not copy and paste, but read first, reflect and only if you understand it execute it. We know we say this all the time, but better saying it again instead of you destroying your system. This command also requires sudo access so you will either have to be in the sudo group, or use

```
1 su <your administrator name>
```

login and then execute the command under root.

```
1 sudo dd if=ubuntu-17.10.1-desktop-amd64.img.dmg of=/dev/diskN bs=1m
```

(Not tested: Using /dev/rdisk instead of /dev/disk may be faster according to the ubuntu documentation)

Ubuntu’s Web page also gives the following tips:

- “If you see the error dd: Invalid number ‘1m’, you are using GNU dd. Use the same command but replace bs=1m with bs=1M.”
- “If you see the error dd: /dev/diskN: Resource busy, make sure the disk is not in use. Start Disk Utility.app and unmount the volume (don’t eject).”

You will see an error window popping up telling you: **The disk inserted was not readable by this compute**. Please, leave the window as is and instead type in on the terminal.

```
1 diskutil eject /dev/diskN
```

Now remove the flash drive, and press in the error window **Ignore**

Now you have a flash drive with ubuntu installed and you can boot from it. To do so, please

**restart your Mac and press option key**

while the Mac is restarting to choose the USB-Stick

You will need a plug for USB keyboard, USB mouse, and network cable.

There are some issue from this point on.

```
1 sudo apt-get update
```

Add universe to the window for application updates

see <https://help.ubuntu.com/community/Repositories/Ubuntu>

```
| sudo apt-get install vnc4server
```

Start the server and set up a password

```
| vncserver
```

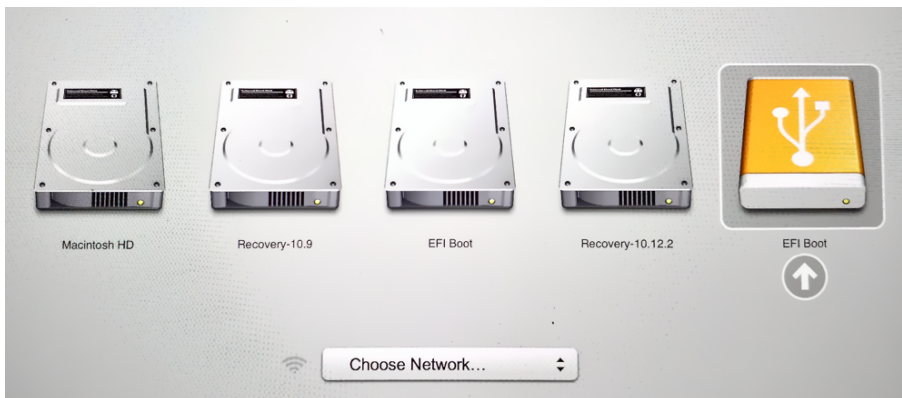
### Warning

The next section is untested and needs verification.

## 81.7.2 Boot from the USB Stick

To boot from the USB stick, you need to restart or power-on the Mac with the USB stick inserted while you press the Option/alt key.

The launch *Startup Manager* will be started showing a list of bootable devices connected to the machine. Your USB stick should appear as gold/yellow and labelled *EFI Boot*. Use your cursor keys to move to the most right EFI boot device in that list (likely the USB stick) and press ENTER. YOU can also use the mouse.



A boot menu will shortly start up and after you press again ENTER your machine will boot into Ubuntu.

For more information on how to setup ubuntu see:

- <https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0>

After you have booted and looged in, you need to update the distribution. We recommend that you switch on Universe in the applications settings.

Next you need to issue in the command terminal

```
| sudo apt-get update
```

You will likely see some warnings with number 95 which you can ignore. Please report your experience and we update this page based on your feedback.

## 81.7.3 Ubuntu on a USB stick for OSX via GUI

An alternative to the Command Line solution to create an USB stick with bootable UBuntu on is to use the OSX GUI. This method is more complex than the command line solution. In addition

as we are learning about cloud computing in this book, it is of advantage to learn how to do this from commandline as the replication of the approach via commandline is easier and more scalable. However for completeness, we have also included here the GUI-based method.

The material in this section was copied and modified from

- <https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-macos>

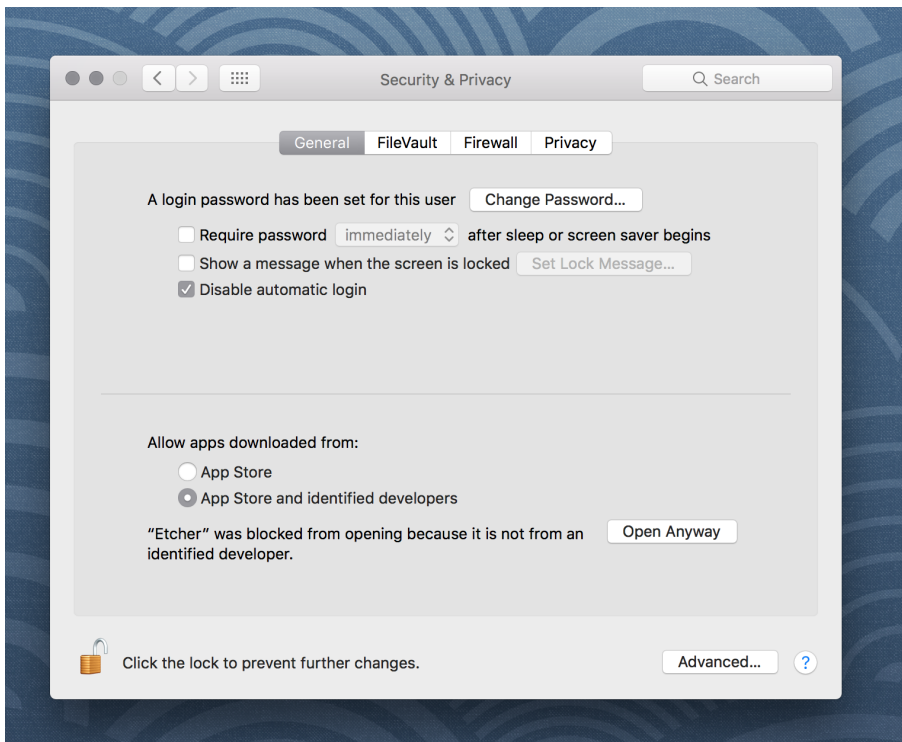
You will need a USB stick/flash drive. We recommend a 8GB or larger. Please let us know if it works for you on larger than 8GB drives.

### Install Etcher

Etcher is a tool that allows you to easily write an ISO onto a USB stick. Etcher is integrated in the OSX GUI environment and allows to drag the iso into it for burning. Etcher can be found at

- <https://etcher.io/>

As this is an application from unidentified developers (not registered in the apple store), you need to enable it after downloading. To do so, you can enable the *App Store and identified developers* in the *Security and Privacy* pane in the System Preferences. IN case you get a warning about running the application, click *Open Anyway* in the same pane.



### Prepare the USB stick

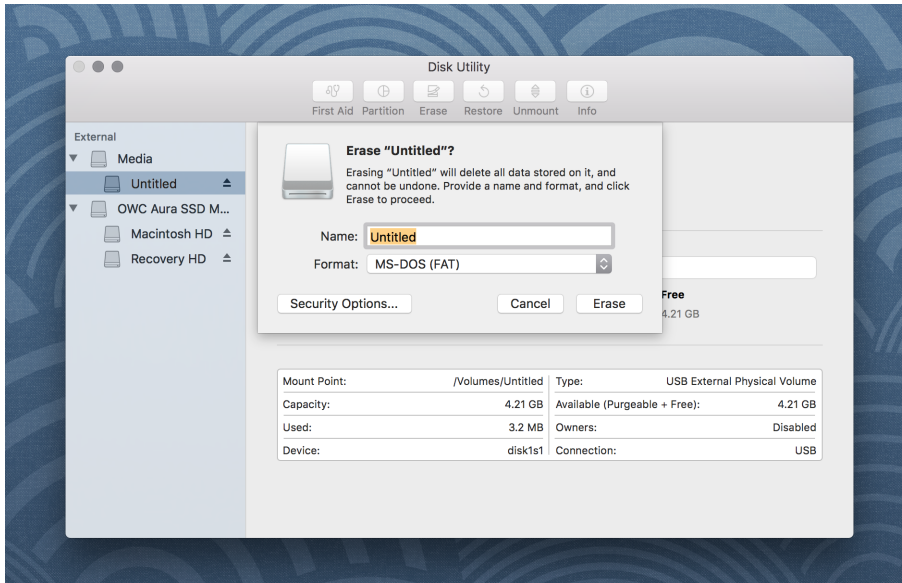
#### Warning

The Disk Utility needs to be used with caution as selecting the wrong device or partition can result in data loss.

Next you need to conduct the following steps which we copied from the Ubuntu Web page:

- Launch Disk Utility from Applications>Utilities or Spotlight search
- Insert your USB stick and observe the new device added to Disk Utility

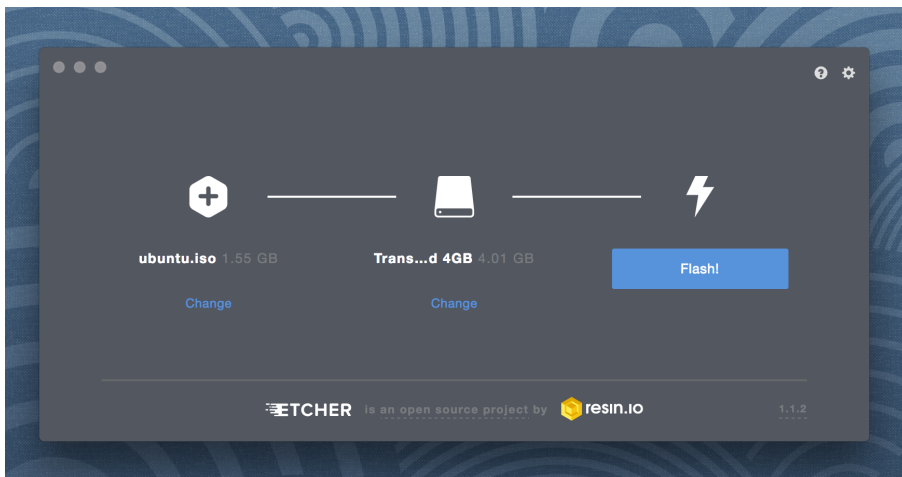
- Select the USB stick device and select Erase from the tool bar (or right-click menu)
- Set the format to MS-DOS (FAT) and the scheme to GUID Partition Map Check you've chosen the correct device and click Erase



## Etcher configuration

Next we use Etcher to configure and write to your USB device as follows (copied from the Ubuntu Web page):

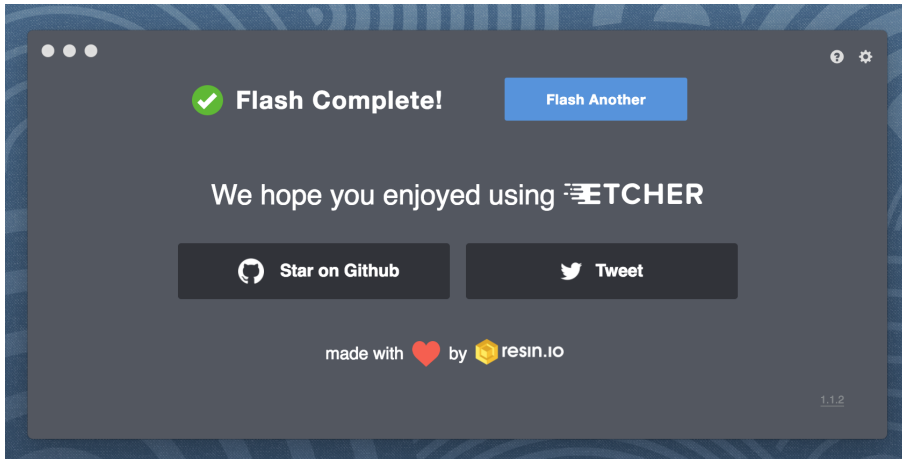
- Select image will open a file requester from which should navigate to and select the ISO file downloaded previously. By default, the ISO file will be in your Downloads folder.
- Select drive, replaced by the name of your USB device if one is already attached, lets you select your target device. You will be warned if the storage space is too small for your selected ISO.
- Flash! will activate when both the image and the drive have been selected. As with Disk Utility, Etcher needs low-level access to your storage hardware and will ask for your password after selection.



### Write to the USB stick

When writing to the USB, Etcher will ask you for your password. It will write the ISO file, once you confirmed the password.

You will see the progress reported to the Etcher window. Once it has finished, Etcher will report on the successful process.



#### Warning

After the write process has completed, macOS may inform you that *\*The disk you inserted was not readable by this computer\**. Don't select Initialise. Instead, select Eject and remove the USB device.



# XIV

Draft: PI



## 81.8 Pi Cluster Related Information

Assignmnet tasks from cluster build on 2/9/2018 ##OSX \* Yue Guo sp18-508 \* Min Chen sp18-405  
##Windows 10 \* Keerthi sp18-602  
##Linux \* Arnov sp18-503  
##ssh without monitor \* Goutham sp18-401  
##User password without monitor \* Surya sp18-418  
##ssh-key \* Priyadarshini sp18-421 \* Ankita sp18-502  
##Docker \* Arnov sp18-503 \* Ardi sp18-411  
##Spark \* Karan Kotabagi sp18-412 \* Manoj \* Min sp18-405 \* Ramya sp18-406 \* Karan Kamatagi sp18-410  
##Kubernetes \* Tim sp18-526 \* Juliano sp18-601  
##back-up SD cards \* Yue sp18-508  
##Duplication \* Keerthi sp18-602  
##Docker Swarm \* Hao Tian sp18-524 \* Lin Qingyun sp18-515





# Draft: Big Data Algorithms

|           |                                                                 |            |
|-----------|-----------------------------------------------------------------|------------|
| <b>82</b> | <b>Technology Training - kNN and Clustering</b>                 | <b>865</b> |
| 82.1      | Recommender Systems - K-Nearest Neighbors                       |            |
| 82.2      | Clustering and heuristic methods                                |            |
| 82.3      | Resources                                                       |            |
| <b>83</b> | <b>Technology for Big Data Applications and Analytics</b>       | <b>869</b> |
| 83.1      | Technology: K-means                                             |            |
| 83.2      | Technology: MapReduce                                           |            |
| 83.3      | Technology: Kmeans and MapReduce Parallelism                    |            |
| 83.4      | Technology: PageRank                                            |            |
| <b>84</b> | <b>Technology Training - Plotviz</b>                            | <b>873</b> |
| 84.1      | Using Plotviz Software for Displaying Point Distributions in 3D |            |





## 82. Technology Training - kNN and Clustering

This section is meant to provide a discussion on the  $k$ th Nearest Neighbor (kNN) algorithm and clustering using K-means. Python version for kNN is discussed in the video and instructions for both Java and Python are mentioned in the slides. Plotviz is used for generating 3D visualizations.

### 82.1 Recommender Systems - K-Nearest Neighbors

We discuss simple Python  $k$  Nearest Neighbor code and its application to an artificial data set in 3 dimensions. Results are visualized in Matplotlib in 2D and with Plotviz in 3D. The concept of training and testing sets are introduced with training set pre-labelled.

Files:

- `kNN.py` </files/python/knn/kNN.py>
- `kNN_Driver.py` </files/python/knn/kNN\_Driver.py>
- `DatingTesting2.txt` </files/python/knn/dating\_test\_set2.txt>
- `clusterFinal-M3-C3Dating-ReClustered.pviz` </files/python/knn/clusterFinal-M3-C3Dating-ReClustered.pviz>
- `DatingRating-OriginalLabels.pviz` </files/python/knn/dating\_rating\_original\_labels.pviz>
- `clusterFinal-M30-C28.pviz` </files/python/knn/clusterFinal-M30-C28.pviz>

#### 82.1.1 Python $k$ 'th Nearest Neighbor Algorithms

This lesson considers the Python  $k$  Nearest Neighbor code found on the web associated with a book by Harrington on Machine Learning. There are two data sets. First we consider a set of 4 2D vectors divided into two categories (clusters) and use  $k=3$  Nearest Neighbor algorithm to classify 3 test points. Second we consider a 3D dataset that has already been classified and show how to normalize. In this lesson we just use Matplotlib to give 2D plots.



### 82.1.2 3D Visualization

The lesson modifies the online code to allow it to produce files readable by PlotViz. We visualize already classified 3D set and rotate in 3D.

### 82.1.3 Testing k'th Nearest Neighbor Algorithms

The lesson goes through an example of using k NN classification algorithm by dividing dataset into 2 subsets. One is training set with initial classification; the other is test point to be classified by k=3 NN using training set. The code records fraction of points with a different classification from that input. One can experiment with different sizes of the two subsets. The Python implementation of algorithm is analyzed in detail.

## 82.2 Clustering and heuristic methods

We use example of recommender system to discuss clustering. The details of methods are not discussed but k-means based clustering methods are used and their results examined in Plotviz. The original labelling is compared to clustering results and extension to 28 clusters given. General issues in clustering are discussed including local optima, the use of annealing to avoid this and value of heuristic algorithms.

Files:

- [Fungi\\_LSU\\_3\\_15\\_to\\_3\\_26\\_zeroidx.pviz </files/python/plotviz/fungi\\_lsu\\_3\\_15\\_to\\_3\\_26\\_zeroidx.pviz>](#)
- [DatingRating-OriginalLabels.pviz </files/python/plotviz/datingrating\\_originallabels.pviz>](#)
- [clusterFinal-M30-C28.pviz </files/python/plotviz/clusterFinal-M30-C28.pviz>](#)
- [clusterFinal-M3-C3Dating-ReClustered.pviz </files/python/plotviz/clusterfinal\\_m3\\_c3dating\\_reclustered.pviz>](#)

### 82.2.1 Kmeans Clustering

We introduce the k means algorithm in a gentle fashion and describes its key features including dangers of local minima. A simple example from Wikipedia is examined.

### 82.2.2 Clustering of Recommender System Example

Plotviz is used to examine and compare the original classification with an *optimal* clustering into 3 clusters using a fancy deterministic annealing method that is similar to k means. The new clustering has centers marked.

### 82.2.3 Clustering of Recommender Example into more than 3 Clusters

The previous division into 3 clusters is compared into a clustering into 28 separate clusters that are naturally smaller in size and divide 3D space covered by 1000 points into compact geometrically local regions.

### 82.2.4 Local Optima in Clustering

This lesson introduces some general principles. First many important processes are *just* optimization problems. Most such problems are rife with local optima. The key idea behind annealing to avoid local optima is described. The pervasive greedy optimization method is described.

### 82.2.5 Clustering in General

The two different applications of clustering are described. First find geometrically distinct regions and secondly divide spaces into geometrically compact regions that may have no *thin air* between them. Generalizations such as mixture models and latent factor methods are just mentioned. The important distinction between applications in vector spaces and those where only inter-point distances are defined is described. Examples are then given using PlotViz from 2D clustering of a mass spectrometry example and the results of clustering genomic data mapped into 3D with Multi Dimensional Scaling MDS.

### 82.2.6 Heuristics

Some remarks are given on heuristics; why are they so important why getting exact answers is often not so important?

TODO: These resources have not all been checked to see if they still exist this is currently in progress

## 82.3 Resources

- <https://en.wikipedia.org/wiki/Kmeans>
- [http://grids.ucsf.edu/ptliupages/publications/DACIDR\\_camera\\_ready\\_v0.3.pdf](http://grids.ucsf.edu/ptliupages/publications/DACIDR_camera_ready_v0.3.pdf)
- <http://salsahpc.indiana.edu/millionseq/>
- <http://salsafungiphy.blogspot.com/>
- <https://en.wikipedia.org/wiki/Heuristic>





## 83. Technology for Big Data Applications and Analysis


We use the K-means Python code in SciPy package to show real code for clustering. After a simple example we generate 4 clusters of distinct centers and various choice for sizes using Matplotlib for visualization. We show results can sometimes be incorrect and sometimes make different choices among comparable solutions. We discuss the *hill* between different solutions and rationale for running K-means many times and choosing best answer. Then we introduce MapReduce with the basic architecture and a homely example. The discussion of advanced topics includes an extension to Iterative MapReduce from Indiana University called Twister and a generalized Map Collective model. Some measurements of parallel performance are given. The SciPy K-means code is modified to support a MapReduce execution style. This illustrates the key ideas of mappers and reducers. With appropriate runtime this code would run in parallel but here the *parallel* maps run sequentially. This simple 2 map version can be generalized to scalable parallelism. Python is used to Calculate PageRank from Web Linkage Matrix showing several different formulations of the basic matrix equations to finding leading eigenvector. The unit is concluded by a calculation of PageRank for general web pages by extracting the secret from Google.

### 83.1 Technology: K-means

We use the K-means Python code in SciPy package to show real code for clustering. After a simple example we generate 4 clusters of distinct centers and various choice for sizes using Matplotlib for visualization. We show results can sometimes be incorrect and sometimes make different choices among comparable solutions. We discuss the *hill* between different solutions and rationale for running K-means many times and choosing best answer.

Files:

[xmean.py](#) 

[sample.csv](#) [parallel-kmeans.py](#) [kmeans-extra.py](#) 

### 83.1.1 K-means in Python

We use the K-means Python code in SciPy package to show real code for clustering and applies it a set of 85 two dimensional vectors -- officially sets of weights and heights to be clustered to find T-shirt sizes. We run through Python code with Matplotlib displays to divide into 2-5 clusters. Then we discuss Python to generate 4 clusters of varying sizes and centered at corners of a square in two dimensions. We formally give the K means algorithm better than before and make definition consistent with code in SciPy.

### 83.1.2 Analysis of 4 Artificial Clusters

We present clustering results on the artificial set of 1000 2D points described in previous lesson for 3 choices of cluster sizes *small large* and *very large*. We emphasize the SciPy always does 20 independent K means and takes the best result -- an approach to avoiding local minima. We allow this number of independent runs to be changed and in particular set to 1 to generate more interesting erratic results. We define changes in our new K means code that also has two measures of quality allowed. The slides give many results of clustering into 2 4 6 and 8 clusters (there were only 4 real clusters). We show that the *very small* case has two very different solutions when clustered into two clusters and use this to discuss functions with multiple minima and a hill between them. The lesson has both discussion of already produced results in slides and interactive use of Python for new runs.

## 83.2 Technology: MapReduce

We describe the basic architecture of MapReduce and a homely example. The discussion of advanced topics includes extension to Iterative MapReduce from Indiana University called Twister and a generalized Map Collective model. Some measurements of parallel performance are given.

### 83.2.1 Introduction

This introduction uses an analogy to making fruit punch by slicing and blending fruit to illustrate MapReduce. The formal structure of MapReduce and Iterative MapReduce is presented with parallel data flowing from disks through multiple Map and Reduce phases to be inspected by the user.

### 83.2.2 Advanced Topics

This defines 4 types of MapReduce and the Map Collective model of Qiu. The Iterative MapReduce model from Indiana University called Twister is described and a few performance measurements on Microsoft Azure are presented.



### 83.3 Technology: Kmeans and MapReduce Parallelism

We modify the SciPy K-means code to support a MapReduce execution style and runs it in this short unit. This illustrates the key ideas of mappers and reducers. With appropriate runtime this code would run in parallel but here the *parallel* maps run sequentially. We stress that this simple 2 map version can be generalized to scalable parallelism.

Files:

[ParallelKmeans](#) 

#### 83.3.1 MapReduce Kmeans in Python

We modify the SciPy K-means code to support a MapReduce execution style and runs it in this short unit. This illustrates the key ideas of mappers and reducers. With appropriate runtime this code would run in parallel but here the *parallel* maps run sequentially. We stress that this simple 2 map version can be generalized to scalable parallelism.

### 83.4 Technology: PageRank

We use Python to Calculate PageRank from Web Linkage Matrix showing several different formulations of the basic matrix equations to finding leading eigenvector. The unit is concluded by a calculation of PageRank for general web pages by extracting the secret from Google.

Files:

[pagerank1.py](#) 

[pagerank2.py](#) 

#### 83.4.1 Calculate PageRank from Web Linkage Matrix

We take two simple matrices for 6 and 8 web sites respectively to illustrate the calculation of PageRank.

#### 83.4.2 Calculate PageRank of a Real Page

This tiny lesson presents a Python code that finds the Page Rank that Google calculates for any page on the web.








## 84. Technology Training - Plotviz


We introduce Plotviz, a data visualization tool developed at Indiana University to display 2 and 3 dimensional data. The motivation is that the human eye is very good at pattern recognition and can *see* structure in data. Although most Big data is higher dimensional than 3, all can be transformed by dimension reduction techniques to 3D. He gives several examples to show how the software can be used and what kind of data can be visualized. This includes individual plots and the manipulation of multiple synchronized plots. Finally, he describes the download and software dependency of Plotviz.

### 84.1 Using Plotviz Software for Displaying Point Distributions in 3D

We introduce Plotviz, a data visualization tool developed at Indiana University to display 2 and 3 dimensional data. The motivation is that the human eye is very good at pattern recognition and can *see* structure in data. Although most Big data is higher dimensional than 3, all can be transformed by dimension reduction techniques to 3D. He gives several examples to show how the software can be used and what kind of data can be visualized. This includes individual plots and the manipulation of multiple synchronized plots. Finally, he describes the download and software dependency of Plotviz.

[Plotvizz \(34\)](#) 

Files:

[Fungi-LSU-3-15-to-3-26-zeroidx.pviz](#) 

[DatingRatings-OriginalLabels.pviz](#) 

[ClusterFinal-M30-C28.pviz](#) 

[section/bigdata/algorithms/plotviz.tex](#)

*clusterFinal-M3-C3Dating-ReClustered.pviz* 

### 84.1.1 Motivation and Introduction to use

The motivation of Plotviz is that the human eye is very good at pattern recognition and can *see* structure in data. Although most Big data is higher dimensional than 3, all data can be transformed by dimension reduction techniques to 3D and one can check analysis like clustering and/or see structure missed in a computer analysis. The motivations shows some Cheminformatics examples. The use of Plotviz is started in slide 4 with a discussion of input file which is either a simple text or more features (like colors) can be specified in a rich XML syntax. Plotviz deals with points and their classification (clustering). Next the protein sequence browser in 3D shows the basic structure of Plotviz interface. The next two slides explain the core 3D and 2D manipulations respectively. Note all files used in examples are available to students.

Video: 7:58: Motivation: <http://youtu.be/4aQlCmQ1jfY>

### 84.1.2 Example of Use I: Cube and Structured Dataset

Initially we start with a simple plot of 8 points -- the corners of a cube in 3 dimensions -- showing basic operations such as size/color/labels and Legend of points. The second example shows a dataset (coming from GTM dimension reduction) with significant structure. This has .pviz and a .txt versions that are compared.

Video: 9:45: Example I: [http://youtu.be/nCTT5mI\\_j\\_Q](http://youtu.be/nCTT5mI_j_Q)

### 84.1.3 Example of Use II: Proteomics and Synchronized Rotation

This starts with an examination of a sample of Protein Universe Browser showing how one uses Plotviz to look at different features of this set of Protein sequences projected to 3D. Then we show how to compare two datasets with synchronized rotation of a dataset clustered in 2 different ways; this dataset comes from k Nearest Neighbor discussion.

Video: 9:14: Proteomics and Synchronized Rotation: <http://youtu.be/lDbIhnLrNkk>

### 84.1.4 Example of Use III: More Features and larger Proteomics Sample

This starts by describing use of Labels and Glyphs and the Default mode in Plotviz. Then we illustrate sophisticated use of these ideas to view a large Proteomics dataset.

Video: 8:37: Larger Proteomics Sample: [http://youtu.be/KBkUW\\_QNSvs](http://youtu.be/KBkUW_QNSvs)

### 84.1.5 Example of Use IV: Tools and Examples

This lesson starts by describing the Plotviz tools and then sets up two examples -- Oil Flow and Trading -- described in PowerPoint. It finishes with the Plotviz viewing of Oil Flow data.

Video: 10:17: Plotviz I: [http://youtu.be/zp\\_709imR40](http://youtu.be/zp_709imR40)

### 84.1.6 Example of Use V: Final Examples

This starts with Plotviz looking at Trading example introduced in previous lesson and then examines solvent data. It finishes with two large biology examples with 446K and 100K points and each with

over 100 clusters. We finish remarks on Plotviz software structure and how to download. We also remind you that a picture is worth a 1000 words.

Video: 14:58: Plotviz II [http://youtu.be/FKoCfTJ\\_cDM](http://youtu.be/FKoCfTJ_cDM)

### 84.1.7 Resources

Download files from <http://salsahpc.indiana.edu/pviz3/>



XXVI

Draft: Artificial Intelligence



# Outdated: Cloud Computing

|           |                                                     |            |
|-----------|-----------------------------------------------------|------------|
| <b>85</b> | <b>OUTDATED: Cloud Computing Fundamentals</b> ..... | <b>883</b> |
| 85.1      | Retired lectures                                    |            |
| 85.2      | Data Center Model                                   |            |
| 85.3      | Data Intensive Sciences                             |            |
| 85.4      | IaaS, PaaS and SaaS                                 |            |
| 85.5      | Challenges                                          |            |
| <b>86</b> | <b>Outdated: IaaS</b> .....                         | <b>885</b> |
| 86.1      | Growth of Virtual Machines                          |            |
| 86.2      | Implementation Levels                               |            |
| 86.3      | Tools and Mechanisms                                |            |
| 86.4      | CPU, Memory & I/O Devices                           |            |
| 86.5      | Clusters and Resource Management                    |            |
| 86.6      | Data Center Automation                              |            |
| 86.7      | Clouds in the Workplace                             |            |
| 86.8      | Checklists and Challenges                           |            |
| 86.9      | Data Center Setup                                   |            |
| 86.10     | Cultivating Clouds                                  |            |











## 85. OUTDATED: Cloud Computing Fundamentals

### 85.1 Retired lectures

*Introduction - (retired) (8:31)* 

*Introduction - (retired) (Page 1)* 

### 85.2 Data Center Model

A look at what truly defines a ‘cloud’. Advantages like scalability and cost-effectiveness have promulgated commercial cloud offerings such as Amazon EC2. Cloud architecture is divided into three layers: Infrastructure as a Service, Platform as a Service, and Software as a Service. AzureBlast is used as an example of how to utilize the cloud setup. Certain misconceptions about clouds are then presented for further discussion.

*Data Center Model (8:08)* 

*Data Center Model (Page 9)* 

### 85.3 Data Intensive Sciences

Some time is spent analyzing the current age of vast data growth, where business, science, and consumer activity has seen an explosion of stored data measured in exabytes. In response to this, the way we conduct scientific research has also undergone an upgrade. However, the average scientist would rather focus on their own research rather than spend time trying to learn different methods of cloud and supercomputing.

*Data Intensive Sciences (2:44)* 

*Data Intensive Sciences (Page 19)* 

## 85.4 IaaS, PaaS and SaaS

Definitions and examples are given for Infrastructure as a Service, Platform as a Service, and Software as a Service. A chart is shown illustrating how use of clouds trades cost and control for efficiency. Following this is an exploration of the MapReduce program, and an illustration of its concepts through WordCount. Finally, four distinct approaches to MapReduce are compared.


*IaaS/PaaS/SaaS (10:17)* 

*IaaS, PaaS and SaaS (Page 25)* 

## 85.5 Challenges

The demands of Big Data calls for advances in areas like distributed computing, systems management, internet technology, and hardware. Clouds have become more prominent in the last few decades, so much so that many people today take advantage of them without even knowing it. Of course, this has also led to increased concerns about security, price, tech support, etc. In spite of this, clouds still have clear advantages over traditional computing models. A quiz is offered at the end asking students to correctly place software in a hierarchy of computing.

*Challenges (5:27)* 

*Challenges (Page 42)* 





## 86. Outdated: IaaS

Examples and definitions are given for SaaS, PaaS, and IaaS. Computational models must be designed with the problems and effective resources in mind. A demonstration of cloud use for Bioinformatics shows how clouds offer advantages of provisioning and virtual cluster support. Overhead and performance issues are touched upon through charts showing the use of three different virtual clusters.

### 86.1 Growth of Virtual Machines

Importance of virtualization is explored, including cross-platform applications. Virtualization has seen rapid growth in recent years in terms of use and services offered. Virtual machines differ from traditional computers in that software virtualization layer (hypervisor) runs on hardware, allowing guest OS to run on top of host OS. VMs can run independent of hardware specifications. Four different types of VM architecture, defined by the layer which the virtual machine monitor (VMM) runs on. VM is identical to physical machines and can be saved and stored, as well as migrated across hardware.

[Growth of Virtual Machines \(10:16\)](#) 

[Growth of Virtual Machines \(Page 28\)](#) 

[Growth of Virtual Machines - pptx \(Page 28\)](#) 

### 86.2 Implementation Levels

Virtualization can be implemented on five levels: application, library, OS, hardware, and instruction. Their benefits are compared in terms of performance, flexibility, complexity, and isolation. A

layout is provided for the Linux virtualization layer, OpenVZ (OS level), which creates virtual private servers. CUDA is a high performance computing library, not designed for VMs; vCUDA is a virtual layer that allows interaction between CUDA and VMs, creating a virtual CUDA library.

[Implementation Levels \(7:57\)](#) 

[Implementation Levels \(Page 41\)](#) 

[Implementation Levels - pptx \(Page 41\)](#) 

### 86.3 Tools and Mechanisms

A list of major hypervisors is given. Type 1 hypervisor resides on the bare metal computer, while Type 2 runs over the host OS. XEN is an open source hardware level hypervisor: consists of hypervisor, kernel, and application. Domain0 in XEN is a VM that manages other VMs. Two types of hardware virtualization: full virtualization and host-based virtualization. Para-virtualization does not need to modify the guest OS like full virtualization and works through hypercalls. An example is the ESX server from VMware.

[Tools and Mechanisms \(7:32\)](#) 

[Tools and Mechanisms \(Page 47\)](#) 

[Tools and Mechanisms - pptx \(Page 47\)](#) 

### 86.4 CPU, Memory & I/O Devices

A hybrid approach to virtualization involves offloading some tasks to the hardware to reduce overhead. This can be combined with para-virtualization for even greater effects. In a guest OS, the VMM provides shadow page tables to transfer virtual memory to machine memory. An example is shown in the Intel Extended Page Table. A virtualization layer for an I/O device is possible, allowing it to act like a physical device and manage host and guest addresses, shown in a detailed VMware example.

[CPU, Memory & I/O Devices \(6:41\)](#) 

[CPU, Memory & I/O Devices \(Page 58\)](#) 

[CPU, Memory & I/O Devices - pptx \(Page 58\)](#) 

### 86.5 Clusters and Resource Management

Characteristics of VM clusters are listed, including the ability to run multiple VMs on the same node and size alteration. Physical clusters are linked through nodes, while virtual clusters can be linked through physical or virtual nodes and can be replicated in virtual servers. Prepackaged OS can be installed in a virtual cluster. Should a VM fail for any reason, its image can be migrated to a new host so work is not lost. An example of this is demonstrated with XEN.

[Clusters and Resource Management \(5:07\)](#) 

[Clusters and Resource Management \(Page 66\)](#) 



[Clusters and Resource Management - pptx \(Page 66\)](#) 

## 86.6 Data Center Automation

Whole data centers can be virtualized, enabling for the construction of private clouds. Some tools for Infrastructure as a Service clouds are Nimbus, Eucalyptus, OpenNebula, and vSphere. Eucalyptus is shown in greater detail. Trust issues in cloud security are answered in virtual machines. Suggested reading material is provided at the end.

[Data Center Automation \(3:30\)](#) 

[Data Center Automation \(Page 74\)](#) 

[Data Center Automation - pptx \(Page 74\)](#) 

## 86.7 Clouds in the Workplace

Clouds run as servers for data storage and sharing on the Internet in an on-demand capacity. Cloud services are scalable depending on the client's needs, allowing for a seemingly limitless source of computing power that can expand or shrink to meet financial demands. Some examples of cloud services are LinkedIn, Amazon S3, and Google App Engine. Different variations of clouds like IaaS and PaaS are offered by both open source and commercial providers. Cloud systems are composed of separate elements like Eucalyptus, Xen and VMware.

[Clouds in the Workplace \(7:13\)](#) 

[Clouds in the Workplace \(Page 1\)](#) 

## 86.8 Checklists and Challenges

The capabilities of several IaaS cloud structures like Amazon EC2 or PaaS like Microsoft Azure are listed. Public and private clouds share certain features; the main difference is public clouds are owned by service providers while private clouds are offered by individual corporations. Certain enabling technologies are required for clouds to provide quick and scalable computing. These include virtual cluster provisioning and multi-tenant environments. PaaS demands the capability to process huge amounts of data as in the case of web searches. Some challenges faced by cloud computing include vendor lock-in owing to lack of standard APIs and metrics; for scientists, there is uncertainty about whether experiments can be reproduced effectively in different cloud environments. However there are distinct advantages clouds potentially have to offer: standardized APIs can eliminate lock-in, and encryption offers data confidentiality.

[Checklists and Challenges \(9:08\)](#) 

[Checklists and Challenges \(Page 11\)](#) 

## 86.9 Data Center Setup

Huge data centers enable cloud computing, containing up to a million servers. Large data centers charge less for their services than small ones. A diagram illustrates the typical setup of a cloud;

rack space on the bottom, on top of which are load balancers, then excess routers and border routers. The next figure compares cost effectiveness in a traditional IT model to a cloud. Other figures display small server clusters and a typical data center arrangement, including emergency power supply and cooling system. A chart shows the power consumption based on CPU, disk, etc. Disks in warehouse servers may be onsite or attached to outside connections like InfiniBand. Switches can form an array of racks. The distribution of memory across a local, rack, or array server in warehouse server setup is listed.

*Data Center Setup (7:49)* 

*Data Center Setup (Page 16)* 

## 86.10 Cultivating Clouds

Power utilization effectiveness (PUE) for a warehouse is determined by comparing it to IT power usage. Racks can contain 40 servers, shipping containers can have up to 1,000 servers; a data center could take 2 years to construct. Warehouse scale computing has greater economy of scale than data centers by reducing network and administrative costs. Individual users can interact with clouds in the SaaS model, while organizations use PaaS. Clouds generally use VMs to recover from system failures. It is predicted that the cloud job market and demand for clouds will experience great growth in the future. Clouds have become ubiquitous in all aspects of the private and public sector. In the future clouds must take into account user privacy, data security and copyright protection.

*Cultivating Clouds (5:10)* 

*Cultivating Clouds (Page 15)* 

*Cultivating Clouds - Conclusions (Page 1)* 

# XVIM Outdated: Data Management

|           |                              |            |
|-----------|------------------------------|------------|
| <b>87</b> | <b>Outdated: NoSQL</b> ..... | <b>891</b> |
| 87.1      | RDBMS vs. NoSQL              |            |
| 87.2      | NoSQL Characteristics        |            |
| 87.3      | BigTable                     |            |
| 87.4      | HBase                        |            |
| 87.5      | HBase Coding                 |            |
| 87.6      | Indexing Applications        |            |
| 87.7      | Related Work                 |            |
| 87.8      | Indexamples                  |            |
| 87.9      | Indexing 101                 |            |
| 87.10     | Social Media Searches        |            |
| 87.11     | Analysis Algorithms          |            |






## 87. Outdated: NoSQL

### 87.1 RDBMS vs. NoSQL

[RDBMS vs. NoSQL \(9:22\)](#) 

[RDBMS vs. NoSQL \(Page 1\)](#) 

[RDBMS vs. NoSQL - pptx \(Page 1\)](#) 

### 87.2 NoSQL Characteristics

Clouds have arisen as an answer to the data demands of social media. Three major programs for NoSQL are BigTable, Dynamo, and CAP theory. NoSQL is not meant to replace SQL, but to tackle the large-data problems SQL is not well equipped to handle. SQL ACID transactions are Atomic, Consistent, Isolated, and Durable. Consistency can be either strong (ACID) or weak (BASE). CAP theorem offers Consistency, Availability, and Partition tolerance, only two of which can coexist for a shared-data system. NoSQL comes in two varieties, each with pros and cons: Key-Value or schema-less. Common advantages of NoSQL include their being open source and fault tolerant.


[NoSQL Characteristics \(10:31\)](#) 

[NoSQL Characteristics \(Page 11\)](#) 

[NoSQL Characteristics - pptx \(Page 11\)](#) 

### 87.3 BigTable

Big Table is a key-value NoSQL model with data arranged in rows and columns. It is composed of Data File System, Chubby, and SSTable. A tablet is a range of rows in BigTable. The master node assigns tablets to tablet servers and manages these servers. Memory is conserved by making SSTables and memtables compact. BigTable is used in features of Google like their search engine and Google Earth.


[BigTable \(6:55\)](#) 

[BigTable \(Page 28\)](#) 


[BigTable - pptx \(Page 28\)](#) 

### 87.4 HBase

HBase is a NoSQL core component of the Hadoop Distributed File System. It is a scalable distributed data store. A timeline of HBase and Hadoop is shown. BigTable still has its uses but does not scale well to large amounts of analytic processing. HBase has a row-column structure similar to BigTable as well as master and slave nodes. Its place in the architecture of HDFS is shown in a diagram.

[HBase \(7:37\)](#) 


[HBase \(Page 44\)](#) 

[HBase - pptx \(Page 44\)](#) 

### 87.5 HBase Coding

This video gives an overview of the code used in the installation of HBase and connecting to it.

[4:30 \(HBase Coding\)](#) 

[HBase Coding \(Page 60\)](#) 

[HBase Coding - pptx \(Page 60\)](#) 

### 87.6 Indexing Applications

A brief summary of the course up to this point is given, followed by a diagram showing the setup of a search engine. Google's search engine contains three key technologies: Google File System, BigTable, and MapReduce. However, research into big data remains difficult owing to the scope of its size. Social media data in particular is a huge source of data with numerous subsets, all of which demands specific approaches in terms of search queries. There are three stages to this approach: query, analysis, and visualization.

[Indexing Applications \(9:33\)](#) 

[Indexing Applications \(Page 1\)](#) 


[Indexing Applications - pptx \(Page 1\)](#) 



## 87.7 Related Work

Indexing improves efficiency in querying data subsets and analysis. Indices can be single (B+, Hash) or multi-dimensional (R, Quad). Four databases which utilize indexing are HBase, Cassandra, Riak, and MongoDB. Current indexing strategies have limits; for instance, they cannot support range queries or only retrieve Top ‘n’ most relevant topics. Customizability of indexing among NoSQL databases is desirable.

*Related Work (5:56)* 


*Related Work (Page 11)* 

*Related Work - pptx (Page 11)* 

## 87.8 Indexexamples

Mapping between metadata and raw index data is the essential issue with indexing. Examples are shown for HBase, Riak, and MongoDB. An abstract index structure contains index keys, entry IDs among multiple entries, and additional fields. Index configuration allows for customizability through choice of fields, which can be anything from timestamps, text, or retweet status.

*Indexexamples (8:35)* 


*Indexexamples (Page 15)* 

*Indexexamples - pptx (Page 15)* 

## 87.9 Indexing 101

User-defined index allows a user to select the fields used in their search. Data records are indexed or un-indexed. Index structure is made up of key, entry ID, and entry fields. A walk-through customized index creation is shown on HBase, called IndexedHBase. HBase is suited to accommodate the creation of index tables. A performance test of IndexedHBase is done on the Truthy Twitter repository, displaying the various tables that can be created with different criteria. Loading time for large-scale historical data can be reduced by adding nodes. Streaming data can be handled by increasing loaders. A comparison of query evaluation is made between IndexedHBase and Riak, with Riak being more efficient with small data loads but IndexedHBase proving superior for large-scale data.

*Indexing 101 (9:53)* 

*Indexing 101 (Page 20)* 

*Indexing 101 - pptx (Page 20)* 

## 87.10 Social Media Searches

The Truthy Project archives social media data by way of metadata memes. Some problems faced in analyzing this data include its large volume, sparsity of information in tweets, and attempting to arrange streaming tweets. Apache Open Stack upgrades Hadoop 2.0 with YARN and a new HDFS. A diagram displays an indexing setup for social media data with YARN.



*Social Media Searches (6:19)* 

*Social Media Searches (Page 28)* 

*Social Media Searches - pptx (Page 28)* 

## 87.11 Analysis Algorithms

Another method of use for inverted indices is in analysis algorithms. The mathematics involved in this is explored, as well as how it relates to index data, mapping, and reducing. Rather than scanning all raw data present, indices allow for searching only the relevant data. An example is given illustrating how this decreases the time needed to search hashtags in Twitter.

*Analysis Algorithms (6:57)* 

*Analysis Algorithms (Page 35)* 

*Analysis Algorithms - pptx (Page 35)* 

XIX

Outdated: SaaS

|           |                                      |            |
|-----------|--------------------------------------|------------|
| <b>88</b> | <b>Outdated: Search Engine</b> ..... | <b>897</b> |
| 88.1      | Google Components                    |            |
| 88.2      | Google Architecture                  |            |
| 88.3      | Google History                       |            |





## 88. Outdated: Search Engine

### 88.1 Google Components

*Google Components (7:02)* 

*Google Components (Page 1)* 

*Google Components - pptx (Page 1)* 

### 88.2 Google Architecture

*Google Architecture (8:40)* 

*Google Architecture (Page 6)* 

*Google Architecture - pptx (Page 6)* 

### 88.3 Google History


Google History: <https://youtu.be/KgONKOXUkHw?t=175> (starting 2:55)

Google Search Engine 1: <https://www.youtube.com/watch?v=S2oT7uMw5Yg>

Google Search Engine 2: <https://www.youtube.com/watch?v=pxos3Yt6y6I>

*Google History (10:36)* 

*Google History (Page 14)* 

*Google History - pptx (Page 14)* 



# Outdated: MapReduce

|           |                                                |            |
|-----------|------------------------------------------------|------------|
| <b>89</b> | <b>Outdated: MapReduce</b> .....               | <b>901</b> |
| 89.1      | Apache Data Analysis OpenStack                 |            |
| 89.2      | MapReduce                                      |            |
| 89.3      | Hadoop Framework                               |            |
| 89.4      | Hadoop Tasks                                   |            |
| 89.5      | Fault Tolerance                                |            |
| 89.6      | Hadoop WordCount on VMs                        |            |
| 89.7      | Programming on a Compute Cluster               |            |
| 89.8      | How Hadoop Runs on a MapReduceJob              |            |
| 89.9      | Literature Review                              |            |
| 89.10     | Introduction to BLAST                          |            |
| 89.11     | BLAST Parallelization                          |            |
| 89.12     | SIMD vs MIMD;SPMD vs MPMD                      |            |
| 89.13     | Data Locality                                  |            |
| 89.14     | Optimal Data Locality                          |            |
| 89.15     | Task Granularity                               |            |
| 89.16     | Resource Utilization and Speculative Execution |            |
| <br>      |                                                |            |
| <b>90</b> | <b>Outdated: Iterative Map Reduce</b> .....    | <b>907</b> |
| 90.1      | Introduction to MapReduce                      |            |
| 90.2      | Google Search Engine 1                         |            |
| 90.3      | Google Search Engine 2                         |            |
| 90.4      | Hadoop PageRank                                |            |
| 90.5      | Discussions and ParallelThinking               |            |
| 90.6      | Hadoop Extensions                              |            |
| 90.7      | Iterative MapReduce Models                     |            |
| 90.8      | Parallel Processes                             |            |
| 90.9      | Static and Variable Data                       |            |
| 90.10     | MapReduce Model Comparison                     |            |
| 90.11     | Twister K-means                                |            |
| 90.12     | Coding and Iterative Alternatives              |            |







## 89. Outdated: MapReduce

### 89.1 Apache Data Analysis OpenStack

The buildup of Big Data has seen the development of new data storage systems like MapReduce and Hadoop. Apache's Big Data Stack houses a host of programs designed around Google's offerings like MapReduce. The architecture of Hadoop 1.0 and 2.0 are compared, along with an examination of the MapReduce concept. A demo video of Twister-MDS includes a 3-dimensional representation of data cluster sorting through the PlotViz program. Data analysis tool Twister boasts features like in-memory support of tasks, data flow separation, and portability.

[Apache Data Analysis OpenStack \(12:01\)](#) 


[Apache Data Analysis OpenStack \(Page 1\)](#) 

[Apache Data Analysis OpenStack - pptx \(Page 1\)](#) 

### 89.2 MapReduce

MapReduce was designed by Google to address the problem of large-scale data processing. A breakdown of basic MapReduce terms and functions follows. Use of MapReduce has flourished since its premier, as illustrated by an in-depth example of its use in WordCount. Finally the basic process of MapReduce is shown.

[MapReduce \(9:07\)](#) 

[MapReduce \(Page 6\)](#) 

[MapReduce - pptx \(Page 6\)](#) 

[section/icloud/course/mapreduce.tex](#)

### 89.3 Hadoop Framework

Hadoop is an open source version of MapReduce designed for broad application in terms of code and settings. Storage is done in the Hadoop Distributed File System through master and slave nodes. Compute is handled by JobTracker and TaskTracker; the duties of these two intertwined programs are then explored more fully.

[Hadoop Framework \(8:32\)](#) 

[Hadoop Framework \(Page 15\)](#) 


[Hadoop Framework - pptx \(Page 15\)](#) 

### 89.4 Hadoop Tasks

The Map stage of MapReduce is shown in greater detail. This process starts with Hadoop Distributed File System, which handles the input data. Key value pairs are assigned to the data blocks. Combiner reduces data size and Partitioner determines distribution of keys among reducers. Intermediate data is stored in a circular buffer before being sent to reduce tasks. Shuffle and Merge are used to order and reduce size of intermediate data. Reduce tasks take over then to determine the output data format. A final chart illustrates the concept of parallelism in MapReduce.

[Hadoop Tasks \(11:01\)](#) 

[Hadoop Tasks \(Page 24\)](#) 


[Hadoop Tasks - pptx \(Page 24\)](#) 

### 89.5 Fault Tolerance

Fault tolerance is a natural benefit of MapReduce. The master node pings worker nodes regularly to verify they are working, and acts accordingly if they do not respond. A diagram illustrates the files which are in charge of things like number of map and reduce tasks, and what to do when the limit is reached on the buffer. The lecture ends with a discussion of class assignments.

[Fault Tolerance \(2:45\)](#) 

[Fault Tolerance \(Page 36\)](#) 

[Fault Tolerance - pptx \(Page 36\)](#) 

### 89.6 Hadoop WordCount on VMs

[Hadoop WordCount on VMs \(7:30\)](#) 

[Hadoop WordCount on VMs \(Page 17\)](#) 

[Hadoop WordCount on VMs - pptx \(Page 17\)](#) 

## 89.7 Programming on a Compute Cluster

Hadoop is now a large part of Yahoo!'s system setup, as well as handling a tremendous variety of data in other areas like medicine and business. A list of time spans for actions in system requirements is given. The original MapReduce was designed to resolve problems like load balancing and machine failures.

[Programming on a Compute Cluster \(6:01\)](#) 

[Programming on a Compute Cluster \(Page 1\)](#) 

[Programming on a Compute Cluster - pptx \(Page 1\)](#) 

## 89.8 How Hadoop Runs on a MapReduceJob

A detailed diagram of the MapReduce job framework is given. This includes task status updates, shuffling, and writing data to nodes. MapReduce is a C++ framework, while Hadoop is written in Java. Shuffling and sorting occurs in the map phase. Reduce reads and writes files to HDFS, and the merger generates the final result. The second Quiz is given at the end.

[How Hadoop Runs on a MapReduceJob \(9:25\)](#) 

[How Hadoop Runs on a MapReduceJob \(Page 8\)](#) 


[How Hadoop Runs on a MapReduceJob - pptx \(Page 8\)](#) 

## 89.9 Literature Review

This video deals primarily with scientific papers written on the topic of MapReduce and related programs. There is a certain criteria for judging scientific submissions. The first paper highlights Google File System, covering topics like data chunks, metadata, and replicas. This is followed by MapReduce and BigTable.

[Literature Review \(9:43\)](#) 

[Literature Review \(Page 16\)](#) 

[Literature Review - pptx \(Page 16\)](#) 

## 89.10 Introduction to BLAST

There are four types of programming model communication patterns: embarrassingly parallel (only map), classic map/reduce, iterative map/reduce, and loosely synchronous. The basic bioinformatics BLAST (Basic Local Alignment Sequence Tool) program data flow is illustrated. An example of database creation comes from the Seattle Children's Hospital. BLAST uses scores to find similar sequences in databases.

[Introduction to BLAST \(8:27\)](#) 

[Introduction to BLAST \(Page 1\)](#) 

[Introduction to BLAST - pptx \(Page 1\)](#) 

### 89.11 BLAST Parallelization

The role of master and worker nodes in BLAST multi-thread usage is discussed. BLAST can be parallelized in several ways: multi-thread, query segmentation, and database segmentation. BLAST is pleasingly parallel in application, but many programs are not. Further information about articles featuring BLAST is provided at the end.

[BLAST Parallelization \(4:44\)](#) 

[BLAST Parallelization \(Page 13\)](#) 

[BLAST Parallelization - pptx \(Page 13\)](#) 

### 89.12 SIMD vs MIMD;SPMD vs MPMD

Four types of parallel models: SISD (traditional PCs), SIMD (GPUs), MISD (shuttle flight control computer), MIMD (distributed systems). Point-to-point (P2P) communication in MPI is used as an example of parallelization. Each successive process adds its own stamp to the data before passing it on to the next. Matrix multiplication for scientific applications differs from the norm in that data is sent in a matrix, not a string. WordCount functions in a map/reduce pattern. These are all types of SIMD. SPMD and MPMD are two other types of model.

[SIMD vs MIMD;SPMD vs MPMD \(9:42\)](#) 

[SIMD vs MIMD;SPMD vs MPMD \(Page 1\)](#) 

[SIMD vs MIMD;SPMD vs MPMD - pptx \(Page 1\)](#) 

### 89.13 Data Locality

A brief review is given of previous topics. As opposed to MPI and HPC, MapReduce brings the computation to the data, rather than vice-versa. This is done to limit energy usage and network congestion. Several factors such as number of nodes and tasks can impact data locality. An equation to improve data locality is tested in an experiment, whose results are given. By default, Hadoop determines scheduling of tasks to available slots in terms of best local composition, not global.

[Data Locality \(8:36\)](#) 

[Data Locality \(Page 10\)](#) 

[Data Locality - pptx \(Page 10\)](#) 

### 89.14 Optimal Data Locality

Global data optimization can be achieved through a proposed algorithm given here. Task, slot, and cost are factors in this algorithm. Network bandwidth must also be taken into consideration when assigning tasks to slots. Linear Sum Assignment Problems require greater time to finish when matrix size is increased. Two different scheduling algorithms were designed to improve the original one in Hadoop. An experiment was run comparing all three, with the network topology-aware algorithm clearly outperforming the others.



[Optimal Data Locality \(4:17\)](#) 

[Optimal Data Locality \(Page 17\)](#) 


[Optimal Data Locality - pptx \(Page 17\)](#) 

## 89.15 Task Granularity

Size of data blocks affects load balancing and overhead. Using Bag of Divisible Tasks method, tasks can be split into sub-tasks and distributed amongst slots to maximize efficiency. When splitting tasks, one must take into account when and which tasks to split, as well as how and how many. In our current proposed algorithm, tasks are split until each slot is occupied. It also uses ASPK (Aggressive Scheduling with Prior Knowledge) to split larger tasks first and when the performance gain is deemed optimal. Optimal and Expected Remaining Job Execution Time can help determine task splitting. Several examples are offered with either single or multiple jobs.

[Task Granularity \(9:51\)](#) 

[Task Granularity \(Page 29\)](#) 

[Task Granularity - pptx \(Page 29\)](#) 

## 89.16 Resource Utilization and Speculative Execution

Resource stealing involves appropriating cores that are kept in reserve on separate nodes and returning them when the computation is over. Speculative execution addresses fault tolerance; when the master node notices a task is running slowly, it will start a speculative task which can take over if it is determined the original task will not finish in time. Overuse of speculative tasks can lead to poor data locality and higher energy demands.

[Resource Utilization and Speculative Execution \(3:52\)](#) 

[Resource Utilization and Speculative Execution \(Page 46\)](#) 

[Resource Utilization and Speculative Execution - pptx \(Page 46\)](#) 





## 90. Outdated: Iterative Map Reduce

### 90.1 Introduction to MapReduce

A review covers cloud computing levels, MapReduce, the course structure, etc. This is followed by a look at Google and their initial offering, Google search engine. Amount of tasks performed on this engine increased considerably over the course of a single decade.

*MapReduce Refresher (9:00)* 

*MapReduce Refresher (Page 1)* 

### 90.2 Google Search Engine 1

The Google web server relies on index and doc servers. Index servers allow the search engine to not have to depend on manually checking every document, reducing computing power demands. Index partitioning can be accomplished either through subsets of documents or words. Basic differences between index and doc servers are discussed. Cache servers save previous query results and can bypass index/doc servers for repeat queries.

*Google Search Engine 1 (8:04)* 

*Google Search Engine 1 (Page 15)* 

*Google Search Engine 1 - pptx (Page 15)* 

*[section/icloud/course/iterative-mapreduce.tex](#)*



### 90.3 Google Search Engine 2

Cache servers greatly enhance the performance of search engines. However, this duplication of queries can lead to higher latency. Crawling in a search engine handles subsets of websites. Batch indexing is the simplest way to create indexes, although it lacks advanced features like checkpointing, which could lead to issues down the line. In-memory index added to Google over a decade ago; increases throughput and decreases latency. Image-based and video-based searches were added in 2007, among others. Google File System, MapReduce and BigTable are key components of Google's current search structure. A discussion of the initial Google proposal paper follows.

[Google Search Engine 2 \(8:32\)](#) 

[Google Search Engine 2 \(Page 21\)](#) 

[Google Search Engine 2 - pptx \(Page 21\)](#) 

### 90.4 Hadoop PageRank

PageRank algorithm in Google ranks a webpage's popularity and relevance. The PageRank calculation formula is examined. After this comes an example of its performance and further mathematical formulae involved in its application.

[Hadoop PageRank \(7:58\)](#) 

[Hadoop PageRank \(Page 1\)](#) 

[Hadoop PageRank - pptx \(Page 1\)](#) 

### 90.5 Discussions and ParallelThinking

Four types of MapReduce: pleasingly parallel, classic, iterative, and loosely synchronous. A diagram shows the flow of data in MapReduce. Specific formulae for PageRank are shown with and without the damping factor. Key-value pairs can be written in matrix form by defining the keys as nodes. Map tasks must make sure to handle dangling nodes (isolated from neighbors), distributed page-rank contribution, and reducer output being the same format as map input. Reduce input is key-value pairs. Ideas behind parallel thinking are analyzed, along with a list of related reading. Seven important questions are asked concerning parallel computing. 13 'Dwarves' are different methods of parallel computing, including MapReduce.

[Discussions and ParallelThinking \(11:12\)](#) 

[Discussions and ParallelThinking \(Page 10\)](#) 

[Discussions and ParallelThinking - pptx \(Page 10\)](#) 

### 90.6 Hadoop Extensions

A model of MapReduce shows its structure. Dryad is Microsoft's version of parallel processing. Twister is an iterative map-reduce framework, as are Haloop, Spark and Pregel. A comparison of their features and capabilities is included.

[Hadoop Extensions \(5:37\)](#) 

[Hadoop Extensions \(Page 50\)](#) 

## 90.7 Iterative MapReduce Models

An introduction to the idea of iterative MapReduce. An overview of other MapReduce models follows. Map Only model has parallel map tasks with no communication between them. Classic MapReduce involves parallel map tasks and reduce tasks which aggregate output and allow legacy code. Loosely Synchronous is an MPI model used in computation and communication of scientific applications.

[Iterative MapReduce Models \(6:46\)](#) 

[Iterative MapReduce Models \(Page 1\)](#) 

[Iterative MapReduce Models - pptx \(Page 1\)](#) 

## 90.8 Parallel Processes

CPU performance increases according to Moore's Law can no longer keep up with the high volume of data being generated. Multi-core architecture is a response to this issue. It requires runtime approaches supporting parallelism, either data-centric for higher throughput (MapReduce) or the traditional HPC approach for optimized computation performance (MPI). MapReduce allows for moving computation to the data. A diagram illustrates the base MapReduce process. MapReduce is designed to improve I/O and handle intermediate data, task scheduling, and fault tolerance. Versions of MapReduce like Hadoop, Dryad and MPI boast different features and programming languages.

[Parallel Processes \(9:44\)](#) 

[Parallel Processes \(Page 4\)](#) 

[Parallel Processes - pptx \(Page 4\)](#) 

## 90.9 Static and Variable Data

Iterative MapReduce was introduced to support high performance systems. It runs iterations of the map/reduce cycles. Data mining algorithms like K-means run numerous iterations. Static data such as data points in K-means does not change, while variable data can alter between each iteration. A naïve iterative MapReduce model can generate huge overhead owing to constantly referencing static data. This can be overcome with long-running map/reduce tasks that distinguish between static and variable data. You can also accelerate the intermediate data transfer or combine the output of all reduce tasks. Iterative MapReduce is shown in the Twister program, which uses the combine output method and determines at the end of every iteration whether to stop or continue with further iterations. The master node in Twister is the Twister Driver, and the slave nodes are Twister Daemons. Twister stores I/O data in partition files. Three MapReduce patterns in Twister: (1) Large input data, reduced in the end; (2) Data size is constant; (3) Data volume increases after MapReduce execution. Data Manipulation Tool handles data loading and uses metadata to keep

track of data in partitions. Twister employs static scheduling. Fault tolerance is reserved for failures that terminate running tasks. Static data can then be used to reassign the failed iterations. A list of Twister APIs is given.

[Static and Variable Data \(11:01\)](#) 

[Static and Variable Data \(Page 10\)](#) 

[Static and Variable Data - pptx \(Page 10\)](#) 

## 90.10 MapReduce Model Comparison

This video showcases examples of work done comparing Twister results with Hadoop, MPI and DryadLINQ. The first is Map Only with CAP3 DNA Sequence Assembly, followed by Classic MapReduce with Pair-wise Sequences and High-Energy Physics, Iterative with K-means clustering, PageRank and Multi-dimensional Scaling, and finally Loosely Synchronous with Matrix Multiplication Algorithms. In all cases, Twister outperforms or is close to the competition.

[MapReduce Model Comparison \(6:56\)](#) 

[MapReduce Model Comparison \(Page 24\)](#) 


[MapReduce Model Comparison - pptx \(Page 24\)](#) 

## 90.11 Twister K-means

Twister is applied to K-means Clustering. K-means develops a set number of clusters by creating cluster centers (centroids) that encompass the data points after successive proximity calculations. Parallelization of K-means is accomplished in the partitions, and the final centroids are determined in the Reduce step. A sample of K-means Clustering code follows, after which Twister is shown being used to determine centroids on K-means. Several questions are posed pertaining to the features of Twister. The results of a Twister K-means run are compared with those from a sequential run. Shown here, as the number of data points increases, Twister's runtimes get progressively faster. In a final set of runs against Hadoop, DryadLINQ, and MPI, Twister outperforms all but MPI.

[Twister K-means \(7:28\)](#) 

[Twister K-means \(Page 34\)](#) 

[Twister K-means - pptx \(Page 34\)](#) 

## 90.12 Coding and Iterative Alternatives

A more detailed look is taken at the code used to run Twister K-means. MapReduce has many programs designed around its setup, including other iterative versions like Haloop, Pregel, and Spark. Twister can extend the use of traditional MapReduce to more complex applications.

[Coding and Iterative Alternatives \(5:14\)](#) 

[Coding and Iterative Alternatives \(Page 43\)](#) 

*Coding and Iterative Alternatives - pptx (Page 43)* 





# Outdated: Internet of Things

|           |                            |            |
|-----------|----------------------------|------------|
| <b>91</b> | <b>Outdated: IoT</b> ..... | <b>915</b> |
| 91.1      | Everyday Data              |            |
| 91.2      | Streaming the Data Ocean   |            |
| 91.3      | Streams of Events          |            |
| 91.4      | Faults & Frameworks        |            |
| 91.5      | Spouts to Bolts            |            |







## 91. Outdated: IoT

### 91.1 Everyday Data

Ph.D. candidate Supun Kamburugamuva goes over the so-called Internet of Things as well as strategies and tools developed for Distributed Stream Processing.

[Everday Data \(9:31\)](#) 

[Everday Data \(Page 4\)](#) 

### 91.2 Streaming the Data Ocean

Ph.D. candidate Supun Kamburugamuva goes over the so-called Internet of Things as well as strategies and tools developed for Distributed Stream Processing.

[Streaming the Data Ocean \(9:38\)](#) 

[Streaming the Data Ocean \(Page 6\)](#) 

### 91.3 Streams of Events

Ph.D. candidate Supun Kamburugamuva goes over the so-called Internet of Things as well as strategies and tools developed for Distributed Stream Processing.

[Streams of Events \(10:44\)](#) 

[Streams of Events \(Page 1\)](#) 

[section/icloud/course/iot.tex](#)

## 91.4 Faults & Frameworks

Ph.D. candidate Supun Kamburugamuva goes over the so-called Internet of Things as well as strategies and tools developed for Distributed Stream Processing.

*Faults & Frameworks (7:46)* 

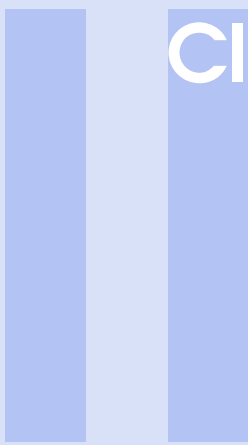
*Faults & Frameworks (Page 9)* 

## 91.5 Spouts to Bolts

Ph.D. candidate Supun Kamburugamuva goes over the so-called Internet of Things as well as strategies and tools developed for Distributed Stream Processing.

*Spouts to Bolts (8:42)* 

*Spouts to Bolts (Page 15)* 



# Class Projects

|           |                           |            |
|-----------|---------------------------|------------|
| <b>92</b> | <b>Projects</b> .....     | <b>919</b> |
| 92.1      | Class: E516               |            |
| 92.2      | Class: E616               |            |
| 92.3      | Class: i524               |            |
| 92.4      | Class: E222               |            |
|           | <b>Bibliography</b> ..... | <b>925</b> |
|           | <b>Index</b> .....        | <b>967</b> |







## 92. Projects

### 92.1 Class: E516

#### 92.1.1 Scope of Project

The objective of the project is to define a clear problem statement and create a framework to address that problem. This framework must include a docker packaged service that includes all necessary dependencies necessary to perform the analysis. For example if you are using a machine learning algorithm your docker packaged service will include the machine learning algorithm which works on a particular dataset providing clustered ,classified or regression outputs to the user. The final objective is to obtain results in form of graphs or tabular mode. In selecting a dataset, you can use a dataset from a public dataset like

- <https://archive.ics.uci.edu/ml/datasets.html>

and pick a suitable dataset according to your problem statement. Final results must be exposed via a REST service. For instance if it is a classification problem, the should have the capability of entering a new test data set or single data point (meaning a single record) via a REST API endpoint and get the expected output in form of a JSON object. UI creation is an optional task. This is the overall expectation of the project.

#### 92.1.2 Deliverables

- Finding a Data Set Size(A good judgement of the datasize would be  $100 \text{ MB} < \text{datasize} < 500\text{MB}$ ): (Estimated Time 1 hour)
- Cleaned Up Data Set: (Estimated Time 0.5 Week)
- Spark Mlib or Scikit Learn ML Algorithm Application: (Estimated Time 0.5 Week) (You can use any other library you prefer)
- Clustering or Regression or Classification results in Graphical Format (Matplotlib) (GUI optional, save graphs as png). Otherwise you can do some analysis on the dataset the way you

propose to do and this way must be explicitly discussed with Professor Gregor. : (Estimated Time 5 days)

- Swagger or Flask Rest Service to send a sample data set and get output (terminal output is enough): (Estimated Time 1 day) (Completing with Swagger Bonus points)
- Take results in a constant environment (ex: Chameleon Cloud or your PC) provide system configuration: (Estimated Time 2 days)
- Create a Makefile with running, setting up, sample test case running commands: (Estimated Time 1 hour)
- requirement.txt file for pip installation: (Estimated Time < 1 hour)
- Package with Docker: (Estimated Time 10 hour)
- 8 Page Report including (Abstract, Introduction, Project Procedure, Technology Usage, Results(Training Time breakdown, Data cleaning time breakdown, system set up time, time taken to system launch, define the performance of the machine you're taking results), Conclusion ,Work Distribution (if you have team members): (40 hours)
- Dockerfile to run your project. (Your project must run in the docker environment, working project within docker is required).

Total Estimated Time : 3 weeks.

### Grading Scheme

- Working Project Inside Docker + Dockerfile : 60%
- Project Report : 40%

## 92.1.3 Helpful Chapters

You can learn how to use docker, referring to section 38. Using docker in Chameleon clouds can be learn via section 39. Need more clarification regarding steps in data cleaning and data processing, refer the section ??.

## 92.2 Class: E616

### 92.2.1 Scope of Project

The objective of the project is to define a clear problem statement and create a framework to address that problem. This framework must include a docker packaged service that includes all necessary dependencies necessary to perform the analysis. For example if you are using a machine learning algorithm your docker packaged service will include the machine learning algorithm which works on a particular dataset providing clustered ,classified or regression outputs to the user. The final objective is to obtain results in form of graphs or tabular mode. In selecting a dataset, you can use a dataset from a public dataset like

- <https://archive.ics.uci.edu/ml/datasets.html>

and pick a suitable dataset according to your problem statement. Final results must be exposed via a REST service. For instance if it is a classification problem, the should have the capability of entering a new test data set or single data point (meaning a single record) via a REST API endpoint and get the expected output in form of a JSON object. UI creation is an optional task. This is the overall expectation of the project.

### 92.2.2 Deliverables

- Finding a Data Set Size(A good judgement of the datasize would be 100 MB < datasize < 500MB): (Estimated Time 1 hour)
- Cleaned Up Data Set: (Estimated Time 0.5 Week)
- Spark Mlib or Scikit Learn ML Algorithm Application: (Estimated Time 0.5 Week) (You can use any other library you prefer)
- Clustering or Regression or Classification results in Graphical Format (Matplotlib) (GUI optional, save graphs as png). Otherwise you can do some analysis on the dataset the way you propose to do and this way must be explicitly discussed with Professor Gregor. : (Estimated Time 5 days)
- Swagger or Flask Rest Service to send a sample data set and get output (terminal output is enough): (Estimated Time 1 day) (Completing with Swagger Bonus points)
- Take results in a constant environment (ex: Chameleon Cloud or your PC) provide system configuration: (Estimated Time 2 days)
- Create a Makefile with running, setting up, sample test case running commands: (Estimated Time 1 hour)
- requirement.txt file for pip installation: (Estimated Time < 1 hour)
- Package with Docker: (Estimated Time 10 hour)
- 8 Page Report including (Abstract, Introduction, Project Procedure, Technology Usage, Results(Training Time breakdown, Data cleaning time breakdown, system set up time, time taken to system launch, define the performance of the machine you're taking results), Conclusion ,Work Distribution (if you have team members): (40 hours)
- Dockerfile to run your project. (Your project must run in the docker environment, working project within docker is required).

Total Estimated Time : 3 weeks.

### 92.2.3 Helpful Chapters

You can learn how to use docker, referring to section 38. Using docker in Chameleon clouds can be learn via section 39. Need more clarification regarding steps in data cleaning and data processing, refer the section ??.

#### Grading Scheme

- Working Project Inside Docker + Dockerfile : 60%
- Project Report : 40%

## 92.3 Class: i524

### 92.3.1 Scope of Project

The objective of the project is to define a clear problem statement and create a framework to address that problem. This framework must include a docker packaged service that includes all necessary dependencies necessary to perform the analysis. For example if you are using a machine learning algorithm your docker packaged service will include the machine learning algorithm which works on a particular dataset providing clustered ,classified or regression outputs to the user. The final objective is to obtain results in form of graphs or tabular mode. In selecting a dataset, you can use a dataset from a public dataset like

- <https://archive.ics.uci.edu/ml/datasets.html>



and pick a suitable dataset according to your problem statement. Final results must be exposed via a REST service. For instance if it is a classification problem, the should have the capability of entering a new test data set or single data point (meaning a single record) via a REST API endpoint and get the expected output in form of a JSON object. UI creation is an optional task. This is the overall expectation of the project.

### 92.3.2 Deliverables

- Finding a Data Set Size(A good judgement of the datasize would be  $100 \text{ MB} < \text{datasize} < 500\text{MB}$ ): (Estimated Time 1 hour)
- Cleaned Up Data Set: (Estimated Time 0.5 Week)
- Spark Mllib or Scikit Learn ML Algorithm Application: (Estimated Time 0.5 Week) (You can use any other library you prefer)
- Clustering or Regression or Classification results in Graphical Format (Matplotlib) (GUI optional, save graphs as png). Otherwise you can do some analysis on the dataset the way you propose to do and this way must be explicitly discussed with Professor Gregor. : (Estimated Time 5 days)
- Swagger or Flask Rest Service to send a sample data set and get output (terminal output is enough): (Estimated Time 1 day) (Completing with Swagger Bonus points)
- Take results in a constant environment (ex: Chameleon Cloud or your PC) provide system configuration: (Estimated Time 2 days)
- Create a Makefile with running, setting up, sample test case running commands: (Estimated Time 1 hour)
- requirement.txt file for pip installation: (Estimated Time < 1 hour)
- Package with Docker: (Estimated Time 10 hour)
- 8 Page Report including (Abstract, Introduction, Project Procedure, Technology Usage, Results(Training Time breakdown, Data cleaning time breakdown, system set up time, time taken to system launch, define the performance of the machine you're taking results), Conclusion ,Work Distribution (if you have team members): (40 hours)
- Dockerfile to run your project. (Your project must run in the docker environment, working project within docker is required).

Total Estimated Time : 3 weeks.

### 92.3.3 Helpful Chapters

You can learn how to use docker, referring to section 38. Using docker in Chameleon clouds can be learn via section 39. Need more clarification regarding steps in data cleaning and data processing, refer the section ??.

#### Grading Scheme

- Working Project Inside Docker + Dockerfile : 60%
- Project Report : 40%

### 92.4 Class: E222

For the final project in this class you need to do the following.

- Find a dataset : A sample data repository can be found at the link below however you are free to use data from elsewhere, like scikit learn libraries.

- <https://archive.ics.uci.edu/ml/datasets.html>

- Use Scikit Learn Library to run Machine Learning algorithm depending on your problem.
- Do some classifications, clustering or a regression calculation using a machine learning algorithm.
- Use the results from classification to draw charts. You can use matplotlib to do this.
- Use REST api to tun ML algorithm i.e. in k-means the user should be able to change the cluster number (k) through a RESTful service.
- Use Flask Rest API to expose the data to the viewers. So people can send a data set and get the outputs as a json object.

### 92.4.1 Deliverables

- Find and clean up data set
- Scikit Learn ML Algorithm Application to cleaned up data set
- Clustering or Regression or Classification results in Graphical Format (Matplotlib) (GUI not needed, save graphs as png)
- Flask Rest service to tune ML algorithm
- Flask Rest Service to send a sample data set and get output (terminal output is enough)
- Create a Makefile with running, setting up, sample test case running commands
- requirement.txt file for pip installation
- Package with Docker
- Report including (Abstract, Introduction, Project Procedure, Technology Usage, Results, Work Distribution (if you have team members))

The remainder of this semester will be used to complete this project.

### Grading Scheme

- Working Project Inside Docker + Dockerfile : 60%
- Project Report : 40%





## Bibliography

### References

- [1] web page. accessed 2017-02-13. URL: <http://opencv.org/> (cited on page 655).
- [2] web page. accessed 2017-02-13. URL: <http://opencv.org/opencv-3-2.html> (cited on page 655).
- [3] *5 Things to Know about dashDB*. web page. URL: [https://www.ibm.com/developerworks/community/blogs/5things/entry/5\\_things\\_to\\_know\\_about\\_dashdb\\_placeholder?lang=en](https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_dashdb_placeholder?lang=en) (cited on page 704).
- [4] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Technical report. Accessed: 2017-1-24. Cornell University, Mar. 2016 (cited on page 665).
- [5] *About OpenNebula*. URL: <https://opennebula.org/about/technology/> (cited on page 743).
- [6] *About Pivotal Gemfire*. Web Page. Version 9.0.1. Accessed: 2017-01-28. URL: [http://gemfire.docs.pivotal.io/gemfire/getting\\_started/gemfire\\_overview.html](http://gemfire.docs.pivotal.io/gemfire/getting_started/gemfire_overview.html) (cited on page 710).
- [7] Abed Abu-Dbai et al. “Enterprise Resource Management in Mesos Clusters”. In: *Proceedings of the 9th ACM International on Systems and Storage Conference*. SYSTOR '16. Haifa, Israel: ACM, 2016, 17:1–17:1. ISBN: 978-1-4503-4381-7. DOI: [10.1145/2928275.2933272](https://doi.org/10.1145/2928275.2933272). URL: <http://doi.acm.org/10.1145/2928275.2933272> (cited on page 721).
- [8] ACADGILD. *Beginner’s Guide for Impala*. Web page. Online; accessed 7-Apr-2017. Mar. 2016. URL: <https://acadgild.com/blog/beginners-guide-impala/> (cited on page 674).
- [9] ACM, Inc. *Spark SQL: Relational Data Processing in Spark*. Web Page. accessed 2017-02-19. Feb. 2016. URL: <http://dl.acm.org/citation.cfm?id=2742797> (cited on page 705).
- [10] ActiveBPEL. *Communicating with the ActiveBPEL Server Administration Interface via Web Services*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: <http://www.activevos.com>

- [com/content/developers/education/sample\\_active\\_bpel\\_admin\\_api/doc/index.html](http://com/content/developers/education/sample_active_bpel_admin_api/doc/index.html) (cited on page 644).
- [11] *ActiveMQ technology*. webpage. accessed: 2017-02-4. URL: <http://activemq.apache.org/> (cited on page 690).
- [12] Aduna. *RDF components*. Web Page. Accessed: 2017-1-25. URL: <https://www.w3.org/RDF/> (cited on page 715).
- [13] Aduna. *sesame components*. Web Page. Accessed: 2017-1-26. URL: <https://projects.eclipse.org/projects/technology.rdf4j> (cited on page 715).
- [14] Aerobatic. *Aerobatic - Overview*. Web Page. accessed: 2017-01-25. Jan. 2017. URL: <https://www.aerobatic.com/docs/overview/> (cited on page 667).
- [15] *Agave API Home - Features Tab*. webpage. Accessed : 02-04-2017. URL: <https://agaveapi.co/platform/features/> (cited on page 671).
- [16] A. Ailijiang, A. Charapko, and M. Demirbas. “Consensus in the Cloud: Paxos Systems Demystified”. In: *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. Aug. 2016, pages 1–10. DOI: [10.1109/ICCCN.2016.7568499](https://doi.org/10.1109/ICCCN.2016.7568499). URL: <http://ieeexplore.ieee.org/abstract/document/7568499/> (cited on page 750).
- [17] Tyler Akidau et al. “MillWheel: Fault-Tolerant Stream Processing at Internet Scale”. In: *Very Large Data Bases*. 2013, pages 734–746 (cited on page 681).
- [18] Davide Albanese et al. “mlpy: Machine Learning Python”. In: *CoRR abs/1202.6548* (2012). URL: <http://arxiv.org/abs/1202.6548> (cited on page 656).
- [19] *Allegro*. Web Page. Accessed: 2017-1-25. URL: <http://allegrograph.com/> (cited on page 713).
- [20] *Allegrow*. Web Page. Accessed: 2017-1-20. URL: <https://en.wikipedia.org/wiki/AllegroGraph> (cited on page 713).
- [21] Alphabet, Inc. *szl - Overview.wiki*. Code Repository. Online; accessed 30-jan-2017. URL: <https://code.google.com/archive/p/szl/wikis/Overview.wiki> (cited on page 678).
- [22] Alphabet, Inc. Web Page. Online; accessed 25-jan-2017. Jan. 2017. URL: <https://cloud.google.com/> (cited on page 668).
- [23] Amazon. *AWS OpsWorks*. Web Page. accessed 2017-01-25. URL: <https://aws.amazon.com/opsworks/> (cited on page 737).
- [24] Amazon. Web Page. Accessed: 04/2018. 2017. URL: <https://aws.amazon.com> (cited on page 668).
- [25] *Amazon Route 53*. Web Page. Accessed: 2017-02-24. URL: [https://en.wikipedia.org/wiki/Amazon\\_Route\\_53](https://en.wikipedia.org/wiki/Amazon_Route_53) (cited on page 746).
- [26] *Amazon S3*. Web Page. Accessed: 2017-1-27. URL: <https://aws.amazon.com/s3/> (cited on page 728).
- [27] Amazon Web services. *Amazon Redshift Management Services*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: <http://docs.aws.amazon.com/redshift/latest/mgmt/overview.html> (cited on page 677).
- [28] Amazon Web Services, Inc. *AWS Elastic Beanstalk*. Web Page. accessed 2017-02-14. URL: <https://aws.amazon.com/elasticbeanstalk/> (cited on pages 667, 668).
- [29] *Ambari*. Web Page. Accessed: 2017-02-04. URL: <https://ambari.apache.org/> (cited on page 747).
- [30] *Github apache/ambari*. Web Page. Accessed: 2017-02-04. URL: <https://github.com/apache/ambari/> (cited on page 747).
- [31] *Hortonworks Apache Ambari*. Web Page. Accessed: 2017-02-04. URL: <http://hortonworks.com/apache/ambari/> (cited on page 747).



- [32] AMQP. *AMQP is the Internet Protocol for Business Messaging*. Web Page. accessed: 2017-01-31. Jan. 2017. URL: <http://www.amqp.org/about/what> (cited on page 692).
- [33] *An Introduction to Windows Azure BLOB Storage*. Web Page. July 2013. URL: <https://www.simple-talk.com/cloud/cloud-data/an-introduction-to-windows-azure-blob-storage/> (visited on 02/13/2017) (cited on pages 728, 729).
- [34] *Any Analytics, Any Data, Simplified*. webpage. URL: <http://www.pentaho.com/product/product-overview> (cited on page 652).
- [35] Apache. web-page. accessed 2017-02-13. URL: <https://hama.apache.org/> (cited on page 686).
- [36] Apache. *Apache arrow*. Webpage. URL: <http://arrow.apache.org/> (cited on page 754).
- [37] apache. *apache storm*. apache storm website. URL: <http://storm.apache.org/> (cited on page 442).
- [38] *Apache Accumulo User Manual Version 1.8*. accessed 2017-02-26. URL: [https://accumulo.apache.org/1.8/accumulo\\_user\\_manual.html](https://accumulo.apache.org/1.8/accumulo_user_manual.html) (cited on page 711).
- [39] Apache Apex. *Application Developer Guide*. Web-page. URL: [https://apex.apache.org/docs/apex/application\\_development/#application-developer-guide](https://apex.apache.org/docs/apex/application_development/#application-developer-guide) (cited on page 756).
- [40] Apache Apex. *Operator Development Guide*. Web-page. accessed 2017-02-27. URL: [https://apex.apache.org/docs/apex/operator\\_development/](https://apex.apache.org/docs/apex/operator_development/) (cited on page 756).
- [41] *Apache Avro*. webpage. accessed: 2017-02-6. URL: <https://avro.apache.org/docs/current/> (cited on page 751).
- [42] Apache Avro. *Apache Avro 1.8.2 Documentation*. Web Page. accessed: 2018-3-23. Feb. 2017. URL: <http://avro.apache.org/docs/1.8.2/index.html> (cited on page 817).
- [43] Apache Avro. *Apache Avro 1.8.2 Getting Started (Java)*. Web Page. accessed: 2018-3-23. Feb. 2017. URL: <http://avro.apache.org/docs/1.8.2/gettingstartedjava.html> (cited on page 819).
- [44] Apache Avro. *Apache Avro 1.8.2 Getting Started (Python)*. Web Page. accessed: 2018-3-23. Feb. 2017. URL: <http://avro.apache.org/docs/1.8.2/gettingstartedpython.html> (cited on pages 817, 819).
- [45] Apache Avro. *Apache Avro 1.8.2 Hadoop MapReduce guide*. Web Page. accessed: 2018-3-23. Feb. 2017. URL: <http://avro.apache.org/docs/1.8.2/mr.html> (cited on page 819).
- [46] Apache Avro. *Apache Avro 1.8.2 Specification*. Web Page. accessed: 2018-3-23. Feb. 2017. URL: [http://avro.apache.org/docs/1.8.2/spec.html#schema\\_record](http://avro.apache.org/docs/1.8.2/spec.html#schema_record) (cited on page 819).
- [47] *Apache Cassandra*. Web Page. 2016. URL: <http://cassandra.apache.org/> (cited on page 712).
- [48] *Apache Giraph*. web page. URL: <https://giraph.apache.org> (cited on page 686).
- [49] *Apache Giraph Wiki*. web page. URL: [https://en.wikipedia.org/wiki/Apache\\_Giraph](https://en.wikipedia.org/wiki/Apache_Giraph) (cited on page 686).
- [50] Apache Knox. *Apache Knox Home*. Website. Feb. 2017. URL: <https://knox.apache.org/> (cited on page 756).
- [51] *Apache License, Version 2.0*. Web Page. Online; accessed 23-Feb-2017. URL: <https://www.apache.org/licenses/LICENSE-2.0> (cited on page 743).
- [52] *Apache Lucene*. Web Page. Jan. 2017. URL: <http://lucene.apache.org/> (cited on page 705).

- [53] *Apache Nifi (aka HDF) data flow across data center*. Web Page. URL: <https://community.hortonworks.com/articles/9933/apache-nifi-aka-hdf-data-flow-across-data-center.html> (cited on page 651).
- [54] *S4: Distributed Stream Computing Platform*. Web Page. Accessed: 2017-02-24. URL: <http://incubator.apache.org/s4/> (cited on page 680).
- [55] *Apache Samza*. en. Web Page. Page Version ID: 764035647. Feb. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Apache\\_Samza&oldid=764035647](https://en.wikipedia.org/w/index.php?title=Apache_Samza&oldid=764035647) (visited on 02/13/2017) (cited on pages 680, 681).
- [56] *Apache Samza, LinkedIn's Framework for Stream Processing*. Web Page. Jan. 2015. URL: <https://thenewstack.io/apache-samza-linkedins-framework-for-stream-processing/> (visited on 02/13/2017) (cited on page 681).
- [57] Apache Software Foundation. *Apache Incubator*. Web Page. accessed 2017-01-29. URL: <http://incubator.apache.org/> (cited on pages 675, 755).
- [58] Apache Software Foundation. *Apache MRQL*. Web Page. accessed 2017-01-29. Apr. 2016. URL: <https://mrql.incubator.apache.org/> (cited on page 675).
- [59] Apache Software Foundation. *Omid Project Incubation Status*. Web Page. accessed 2017-02-25. Apr. 2016. URL: <https://mrql.incubator.apache.org/> (cited on page 755).
- [60] Apache Software Foundation. *1.1. Document Storage*. Web page. accessed 26-feb-2017. Feb. 2017. URL: <http://docs.couchdb.org/en/stable/intro/overview.html> (cited on page 709).
- [61] Apache Software Foundation. *Apache Phoenix: OLTP and operational analytics for Apache Hadoop*. Web page. Online; accessed 25-jan-2017. Jan. 2017. URL: <http://phoenix.apache.org/> (cited on page 674).
- [62] Apache Software Foundation. *Apache Spark GraphX*. Web Page. accessed 2017-01-30. Feb. 2017. URL: <http://spark.apache.org/graphx/> (cited on page 660).
- [63] *Apache Tajo*. Web Page. Accessed: 2017-1-27. URL: <https://tajo.apache.org> (cited on page 673).
- [64] *Apache Tajo Quick Guide*. Web Page. Accessed: 2017-1-25. URL: [https://www.tutorialspoint.com/apache\\_tajo/apache\\_tajo\\_quick\\_guide.htm](https://www.tutorialspoint.com/apache_tajo/apache_tajo_quick_guide.htm) (cited on page 673).
- [65] Apache Tinker Pop Home. *Apache TinkerPop Home*. Web Page. 2016. URL: <https://tinkerpop.apache.org/> (cited on page 661).
- [66] *Apache Wink*. Web Page. Accessed: 2017-02-24. URL: <http://wink.apache.org/index.html> (cited on page 755).
- [67] *Apache Hbase*. Web Page. Accessed: 2017-1-26. URL: <https://hbase.apache.org/> (cited on page 710).
- [68] ApacheTinkerPopDoc. *Apache TinkerPop*. Web Page. 2016. URL: <http://tinkerpop.apache.org/docs/3.1.1-incubating/tutorials/getting-started/> (cited on page 661).
- [69] *Apache Sentry Tutorial*. Web Page. Accessed: 2017-02-11. URL: <https://cwiki.apache.org/confluence/display/SENTRY/Sentry+Tutorial> (cited on page 749).
- [70] *Apache NiFi*. Web Page. URL: <https://nifi.apache.org> (cited on page 651).
- [71] *AppEngine - platform as a service*. webpage. Accessed : 02-22-2017. URL: <https://cloud.google.com/appengine> (cited on page 666).
- [72] Wikipedia. *Google App Engine*. webpage. Accessed : 02-22-2017. Jan. 2017. URL: [https://en.wikipedia.org/wiki/Google\\_App\\_Engine](https://en.wikipedia.org/wiki/Google_App_Engine) (cited on page 666).
- [73] appfog. *Overview*. Online. URL: <https://www.ct1.io/appfog/#Overview> (cited on page 670).



- [74] AppScale. *what-is-appscale*. Web Page. 2016. URL: <https://www.appscale.com/community/what-is-appscale/> (cited on page 666).
- [75] Appscale. *why-use-appscale*. Web Page. 2016. URL: <https://www.appscale.com/get-started/deployment-types/> (cited on page 667).
- [76] Thusoo Ashish et al. *Hive - A Petabyte Scale Data Warehouse Using Hadoop*. Paper. Apr. 2010. URL: <http://datamining.uos.ac.kr/wp-content/uploads/2015/07/Hive-A-Petabyte-Scale-Data-Warehouse-Using-Hadoop.pdf/> (cited on page 672).
- [77] Storage Networking Industry Association. *About the SNIA*. Web Page. Accessed: 2017-02-27. URL: <https://www.snia.org/about> (cited on page 731).
- [78] Storage Networking Industry Association. *Cloud Data Management Interface*. 1.1.1. Accessed: 2017-02-27. Storage Networking Industry Association. Colorado Springs, CO, Mar. 2015. URL: [https://www.snia.org/sites/default/files/CDMI\\_Spec\\_v1.1.1.pdf](https://www.snia.org/sites/default/files/CDMI_Spec_v1.1.1.pdf) (cited on page 731).
- [79] Storage Networking Industry Association. *Cloud Data Management Interface*. Web Page. Accessed: 2017-02-27. Mar. 2015. URL: <https://www.snia.org/cdmi> (cited on page 732).
- [80] *atI*. Web Page. Accessed: 2017-1-22. URL: <http://www.cyverse.org/atmosphere> (cited on page 672).
- [81] Atlassian. *eScience Central Overview*. Web Page. Accessed: 2017-1-24. URL: <https://bitbucket.org/digitalinstitute/esciencecentral/> (cited on page 650).
- [82] Abel Avram. *Phoenix: Running SQL Queries on Apache HBase [Updated]*. Web page. Online; accessed 25-jan-2017. Jan. 2013. URL: <https://www.infoq.com/news/2013/01/Phoenix-HBase-SQL> (cited on page 674).
- [83] Amazon. *AWSLambda*. Web Page. Accessed:2/18/2017. URL: <https://aws.amazon.com/lambda/faqs/> (cited on page 694).
- [84] Amazon. *AWSLambdaEvent*. Web Page. Accessed:2/18/2017. URL: <http://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#intro-core-components-event-sources> (cited on page 694).
- [85] *Get started with Azure Queue storage using .NET*. Web Page. Accessed: 2017-02-10. URL: <https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-queues> (cited on page 694).
- [86] *Github Azure/ azure-stream-analytics*. Web Page. Accessed: 2017-02-03. URL: <https://github.com/Azure/azure-stream-analytics/> (cited on page 683).
- [87] *Microsoft Azure Real-time data analytics*. Web Page. Accessed: 2017-02-03. URL: <https://azure.microsoft.com/en-us/services/stream-analytics/> (cited on page 683).
- [88] Vasavi\*1 B et al. *HIBERNATE TECHNOLOGY FOR AN EFFICIENT BUSINESS APPLICATION EXTENSION*: Paper. June 2011. URL: <https://www.rroij.com/open-access/hibernate-technology-for-an-efficient-business-application-extension-118-125.pdf> (cited on page 698).
- [89] *Background*. web-page. accessed 2017-02-13. URL: <https://orc.apache.org/docs/> (cited on page 718).
- [90] *Riak-KV - NOSQL Key Value Database*. Web Page. Accessed: 2017-01-20. URL: <http://basho.com/products/riak-kv/> (cited on page 706).
- [91] *Riak-TS - NOSQL Time Series Database*. Web Page. Accessed: 2017-01-20. URL: <http://basho.com/products/riak-ts/> (cited on page 706).
- [92] *Riak-S2 - Cloud Object Storage Software*. Web Page. Accessed: 2017-01-20. URL: <http://basho.com/products/riak-s2/> (cited on page 706).

- [93] Kent Baxley, JD la Rosa, and Mark Wenning. “Deploying workloads with Juju and MAAS in Ubuntu 14.04 LTS”. In: *Deploying workloads with Juju and MAAS in Ubuntu 14.04 LTS*. Dell Inc, Technical White Paper, May 2014 (cited on page 735).
- [94] *Berkeley DB Tutorial and Reference Guide*. Web Page. Accessed: 2017-02-11. URL: <https://web.stanford.edu/class/cs276a/projects/docs/berkeleydb/reftoc.html> (cited on page 707).
- [95] *Berkeley DB Wiki*. Web Page. Accessed: 2017-02-11. URL: [https://en.wikipedia.org/wiki/Berkeley\\_DB](https://en.wikipedia.org/wiki/Berkeley_DB) (cited on page 707).
- [96] Swapnil Bhartiya. *How to Use DockerHub*. Blog. accessed: 2018-3-26. Jan. 2018. URL: <https://www.linux.com/blog/learn/intro-to-linux/2018/1/how-use-dockerhub> (cited on pages 497, 500).
- [97] *What is BigQuery? Google Cloud Platform*. Web Page. Accessed: 2017-02-23. URL: <https://cloud.google.com/bigquery> (cited on page 677).
- [98] *What is BigQuery? BigQuery Documentation Google Cloud Platform*. Web Page. Accessed: 2017-02-23. URL: <https://cloud.google.com/bigquery/what-is-bigquery> (cited on page 677).
- [99] Tobias Binz et al. *OpenTOSCA -- A Runtime for TOSCA-Based Cloud Applications*. Edited by Samik Basu et al. Springer Berlin Heidelberg, 2013, pages 692–695. ISBN: 978-3-642-45005-1. URL: [http://dx.doi.org/10.1007/978-3-642-45005-1\\_62](http://dx.doi.org/10.1007/978-3-642-45005-1_62) (cited on page 738).
- [100] *About Bioconductor*. Web Page. Accessed: 2017-02-10. URL: <https://www.bioconductor.org/about/> (cited on page 654).
- [101] bioKepler. *Demo Workflow*. WebPage. URL: <http://www.biokepler.org/userguide#demos> (cited on page 646).
- [102] bioKepler. *What is bioKepler*. WebPage. URL: <http://www.biokepler.org/faq#what-is-biokepler> (cited on page 646).
- [103] Bipin. *Difference between vSphere, ESXi and vCenter*. Webpage. Aug. 2012. URL: <http://www.mustbegeek.com/difference-between-vsphere-esxi-and-vcenter/> (cited on page 745).
- [104] *BITTORRENT*. Web Page. 2017. URL: <https://www.lifewire.com/how-torrent-downloading-works-2483513> (cited on page 719).
- [105] Oscar Boykin et al. *Summingbird*. Website. README.md, github. URL: <https://github.com/twitter/summingbird> (cited on page 679).
- [106] Oscar Boykin et al. “Summingbird: A framework for integrating batch and online mapreduce computations”. In: *Proceedings of the VLDB Endowment* 7.13 (2014), pages 1441–1451 (cited on page 679).
- [107] *Business Process Execution Language*. Web Page. Accessed: 2017-02-11. URL: [https://en.wikipedia.org/wiki/Business\\_Process\\_Execution\\_Language](https://en.wikipedia.org/wiki/Business_Process_Execution_Language) (cited on page 643).
- [108] Mikio L. Braun. *Magastore, Spanner - distributed databases*. Web Page. Accessed: 2017-01-28. Nov. 2013. URL: <http://blog.mikiobraun.de/2013/03/more-google-papers-magastore-spanner-voted-commits.html> (cited on page 711).
- [109] Nathan Bronson and Inc. others Facebook. “TAO: Facebook’s Distributed Data Store for the Social Graph”. In: *2013 USENIX Annual Technical Conference*. 2013. URL: <http://ai2-s2-pdfs.s3.amazonaws.com/39ac/2e0fc4ec63753306f99e71e0f38133e58ead.pdf> (cited on page 714).
- [110] Martin Brown. *The Technology Behind Couchbase*. Web page. Online; accessed 29-jan-2017. Mar. 2012. URL: <https://www.safaribooksonline.com/blog/2012/03/01/the-technology-behind-couchbase/> (cited on page 709).

- [111] Jason Brownlee. *A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library*. webpage. Apr. 16, 2014. URL: <http://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/> (cited on page 657).
- [112] Alfredo Buttari et al. “A class of parallel tiled linear algebra algorithms for multicore architectures”. In: *Parallel Computing* 35.1 (2009), pages 38–53. URL: <http://www.sciencedirect.com/science/article/pii/S0167819108001117> (visited on 02/13/2017) (cited on page 655).
- [113] Byung-Gon Chun. *REEFProposal - Incubator*. Web Page. accessed 2017-01-28. Aug. 2014. URL: <https://wiki.apache.org/incubator/ReefProposal> (cited on page 685).
- [114] AT&T Laboratories Cambridge. *The Medusa Applications Environment*. Web page. Last accessed: 2017.02.25. URL: <http://www.cl.cam.ac.uk/research/dtg/attarchive/medusa.html> (cited on page 688).
- [115] Paul Caponetti. *Why MQTT is the Protocol of Choice for the IoT*. xively.com blog website. Aug. 2017. URL: <http://blog.xively.com/why-mqtt-is-the-protocol-of-choice-for-the-iot/> (cited on page 439).
- [116] *CASCADING*. Web Page. 2017. URL: <http://www.cascading.org/projects/cascading/> (cited on page 649).
- [117] *Cedar stack*. Web Page. Accessed:1/27/2017. URL: <https://devcenter.heroku.com/articles/stack#cedar> (cited on page 667).
- [118] *Celery*. Webpage. URL: <http://www.celeryproject.org/> (cited on page 722).
- [119] *Celery - Distributed Task Queue*. Web Page. URL: <http://docs.celeryproject.org/en/latest/index.html> (cited on page 723).
- [120] Craig Chambers et al. “FlumeJava: Easy, Efficient Data-Parallel Pipelines”. In: *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. Accessed : 04-09-2017. 2 Penn Plaza, Suite 701 New York, NY 10121-0701, 2010, pages 363–375. URL: <http://dl.acm.org/citation.cfm?id=1806638> (cited on page 648).
- [121] Desmond Chan. *Big Data with Apache Apex*. Web-page. accessed 2017-02-27. Jan. 2016. URL: <https://jaxenter.com/big-data-apache-apex-122839.html> (cited on page 756).
- [122] *Chef Commercial Support*. Web Page. URL: <https://www.chef.io/support/> (cited on page 733).
- [123] Guoqiang Jerry Chen et al. “Realtime Data Processing at Facebook”. In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD ’16. San Francisco, California, USA: ACM, 2016, pages 1087–1098. ISBN: 978-1-4503-3531-7. DOI: 10.1145/2882903.2904441. URL: <http://doi.acm.org/10.1145/2882903.2904441> (cited on page 683).
- [124] Yueguo Chen et al. “A Study of SQL-on-Hadoop Systems”. In: *Big Data Benchmarks, Performance Optimization, and Emerging Hardware: 4th and 5th Workshops, BPOE 2014, Salt Lake City, USA, March 1, 2014 and Hangzhou, China, September 5, 2014, Revised Selected Papers*. Springer International Publishing, 2014, pages 154–166. ISBN: 978-3-319-13021-7 (cited on page 676).
- [125] Rudi Cilibrasi et al. *What is CompLearn*. webpage. URL: <http://complearn.org/> (cited on page 657).
- [126] *Cinder - Openstack*. Web Page. Accessed: 2017-1-21. URL: <https://wiki.openstack.org/wiki/Cinder> (cited on page 726).
- [127] *CINET - CyberInfrastructure for Network Science*. Web Page. URL: [www.bi.vt.edu](http://www.bi.vt.edu) (cited on page 662).

- [128] Tait Clarridge. *Disco - A Powerful Erlang and Python Map/Reduce Framework*. Blog, accessed 25-feb-2017. May 2014. URL: <http://www.taitclarridge.com/techlog/2014/05/disco-a-powerful-erlang-and-python-mapreduce-framework.html> (cited on page 686).
- [129] *Cloud and systems management*. webpage. URL: <http://www.cisco.com/c/en/us/products/cloud-systems-management> (cited on page 736).
- [130] Vineet Badola. *Cloud Foundry Blog*. Blog. Sept. 2015. URL: <http://cloudacademy.com/blog/cloud-foundry-benefits/> (cited on page 668).
- [131] Cloudability Inc. *Cloudability Cost Management Tools | Cloudability*. Web Page. Accessed: 2017-02-23. Feb. 2017. URL: <https://www.cloudability.com/product/> (cited on page 753).
- [132] *Cloudbees Wikipedia Documentation*. Web Page. Accessed: 2017-02-13. URL: <https://en.wikipedia.org/wiki/CloudBees> (cited on page 670).
- [133] *Cloudbees Webpage Documentation*. Web Page. Accessed: 2017-02-13. URL: <https://www.cloudbees.com/products> (cited on page 670).
- [134] *OpenStack Keystone Verification*. Web Page. Accessed: 2017-02-12. URL: <https://www.cloudberrylab.com/blog/openstack-keystone-authentication-explained/> (cited on page 748).
- [135] cloudera. *Untangling Apache Hadoop YARN Part 1 Cluster and YARN Basics*. Web Page. 2015. URL: <https://blog.cloudera.com/blog/2015/09/untangling-apache-hadoop-yarn-part-1/> (cited on page 721).
- [136] Cloudera Inc. *Cloudera Impala Overview*. Web page. Online; accessed 7-Apr-2017. URL: [https://www.cloudera.com/documentation/enterprise/5-3-x/topics/impala\\_intro.html](https://www.cloudera.com/documentation/enterprise/5-3-x/topics/impala_intro.html) (cited on page 674).
- [137] cloudmesh. *cloudmesh.pi*. github. ocober 2017. URL: <https://github.com/cloudmesh/cloudmesh.pi> (cited on page 444).
- [138] *CNTK/CNTKBook*. Web Page. URL: <https://github.com/Microsoft/CNTK/blob/master/Documentation/CNTK-TechReport/lyx/CNTKBook-20160217.pdf> (visited on 02/13/2017) (cited on page 666).
- [139] Jeffrey Ira Cohen, Luke Lonergan, and Caleb E. WeltonCohen. *Integrating map-reduce into a distributed relational database*. grant. Dec. 2016. URL: <https://www.google.com/patents/US9514188> (cited on page 720).
- [140] Community Grids Lab IU. *Some of the salient features in naradabrokering*. Web Page. Accessed: 2017-02-15. 501 N. MORTON ST, SUITE 224 BLOOMINGTON IN 47404: Pervasive Technology Labs at Indiana University, Nov. 2009. URL: <http://www.naradabrokering.org/> (cited on page 691).
- [141] Community Grids Lab IU. *The NaradaBrokering Project @ IU Community Grids Laboratory*. Web Page. Accessed: 2017-02-15. 501 N. MORTON ST, SUITE 224 BLOOMINGTON IN 47404: Pervasive Technology Labs at Indiana University, Nov. 2009. URL: <http://www.naradabrokering.org/> (cited on page 691).
- [142] Continuum Analytics. *Blaze*. Web page. accessed 25-feb-2017. 2015. URL: <http://blaze.readthedocs.io/en/latest/index.html#> (cited on page 753).
- [143] Darren Cook. *Practical Machine Learning with H2O*. O'Reilley Media, Dec. 2017, page 300. ISBN: 978-1-4919-6454-5. URL: <https://books.google.com/books?%20id=nJWmDQAAQBAJ&pg=PP2&dq=h2o+software> (cited on page 659).
- [144] Emilio Coppa. *Hadoop Architecture Overview*. Code Repository. accessed 2017-03-23. URL: <http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html> (cited on page 684).

- [145] James C Corbett et al. “Spanner: Google’s globally distributed database”. In: *ACM Transactions on Computer Systems (TOCS)* 31.3 (2013), page 8. URL: [http://dl.acm.org/ft\\_gateway.cfm?id=2491245&type=pdf](http://dl.acm.org/ft_gateway.cfm?id=2491245&type=pdf) (cited on page 711).
- [146] CoreOS. Web Page. Accessed:1/27/2017. URL: <https://www.coreos.com/> (cited on page 744).
- [147] CoreOS. *Why CoreOS*. Web Page. accessed: 2017-01-23. Jan. 2017. URL: <https://coreos.com/why/> (cited on page 744).
- [148] iMatix Corporation. *OMQ - The Guide*. Web Page. Accessed: 2017-1-24. URL: <http://zguide.zeromq.org/page:all> (cited on page 689).
- [149] iMatix Corporation. *Distributed Messaging*. Web Page. Accessed: 2017-1-24. URL: <http://zeromq.org> (cited on page 689).
- [150] Couchbase, Inc. *Couchbase and Apache CouchDB compared*. Web page. accessed 26-feb-2017. Feb. 2017. URL: <https://www.couchbase.com/couchbase-vs-couchdb> (cited on page 709).
- [151] Ian Craggs. *MQTT security: Who are you? Can you prove it? What can you do?* IBM developer works website. Mar. 2013. URL: [https://www.ibm.com/developerworks/community/blogs/c565c720-fe84-4f63-873f-607d87787327/entry/mqtt\\_security?lang=en](https://www.ibm.com/developerworks/community/blogs/c565c720-fe84-4f63-873f-607d87787327/entry/mqtt_security?lang=en) (cited on pages 441, 442).
- [152] Cray Inc. *Cray Yarcdata Urika appliance*. Web Page. 2017. URL: <http://www.cray.com/products/Urika.aspx> (cited on page 713).
- [153] Cray Inc. *Cray Urika-GD Technical Specification*. Technical report. Cray Inc, 2014. URL: <http://www.cray.com/sites/default/files/resources/Urika-GD-TechSpecs.pdf> (cited on page 713).
- [154] CUBRID. Web Page. 2017. URL: <http://www.cubrid.org/> (cited on page 702).
- [155] Baojiang Cui and Tao Xi. “Security analysis of openstack keystone”. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on*. IEEE. 2015, pages 283–288 (cited on page 748).
- [156] Curoverse, Inc. *Arvados | Introduction to Crunch*. Web Page. Accessed: 2017-02-22. URL: <http://doc.arvados.org/user/tutorials/intro-crunch.html> (cited on page 649).
- [157] Curoverse, Inc. *Arvados | Open Source Big Data Processing and Bioinformatics*. Web Page. Accessed: 2017-02-22. 2016. URL: <https://arvados.org> (cited on page 649).
- [158] *D3 Data-Driven Documents*. Web Page. Accessed: 2017-02-11. URL: <https://d3js.org/> (cited on page 665).
- [159] *DataFu*. Web Page. Accessed:1/16/2017. URL: <https://datafu.incubator.apache.org/> (cited on page 653).
- [160] DataNucleus. *DataNucleus Support*. Web Page. Accessed: 2017-02-04. URL: <http://www.datanucleus.com/> (cited on page 699).
- [161] Wikipedia. *IBM DB2-Wikipedia*. Web Page. Online,Accessed: 2017-02-24. Feb. 2017. URL: [https://en.wikipedia.org/wiki/IBM\\_DB2](https://en.wikipedia.org/wiki/IBM_DB2) (cited on page 700).
- [162] *DB2 Introduction*. Web Page. Online,Accessed: 2017-02-24. June 2016. URL: [https://www.tutorialspoint.com/db2/db2\\_introduction.htm](https://www.tutorialspoint.com/db2/db2_introduction.htm) (cited on page 700).
- [163] DC.js. *dc.js - Dimensional Charting Javascript Library*. Web Page. accessed: 2017-01-21. Jan. 2017. URL: <https://dc-js.github.io/dc.js/> (cited on page 665).
- [164] John Denero. *CS61A: Online Textbook*. This book is derived from the classic textbook Structure and Interpretation of Computer Programs by Abelson, Sussman, and Sussman. John Denero originally modified if for Python for the Fall 2011 semester. <http://www-inst.eecs.berkeley.edu/cs61a/sp12/book/>. <http://www-inst.eecs.berkeley.edu/cs61a/sp12/book/>:



- Berkeley, 2011. URL: <http://www-inst.eecs.berkeley.edu/~cs61a/sp12/book/communication.html> (cited on page 662).
- [165] DevTopics. Web Page. URL: <http://www.devtopics.com/kite-obscure-programming-language-of-the-month/> (cited on page 672).
- [166] DeZyre. *How LinkedIn Uses Hadoop To Leverage Big Data Analytics*. Web page. accessed 10-March-2016. This page was last modified on 18 March 2017, at 22:27. URL: <https://www.dezyre.com/article/how-linkedin-uses-hadoop-to-leverage-big-data-analytics/229> (cited on page 682).
- [167] *CDF, Common Data Format (multidimensional datasets)*. Web page. Accessed: 2017-1-28. Mar. 2014. URL: <http://www.digitalpreservation.gov/formats/fdd/fdd000226.shtml#useful> (cited on page 717).
- [168] *Distributed Machine Learning*. Web page. Accessed 25-feb-2017. Feb. 2017. URL: <http://www.mlbase.org/> (cited on page 653).
- [169] Docker. *Overview of Docker Hub*. Web Page. accessed: 2018-3-26. Mar. 2018. URL: <https://docs.docker.com/docker-hub/> (cited on pages 497, 500).
- [170] Docker. *Repositories on Docker Hub*. Web Page. accessed: 2018-3-26. Mar. 2018. URL: <https://docs.docker.com/docker-hub/repos/> (cited on pages 499, 500).
- [171] *Docker Swarm*. WebPage. Accessed: 2017-8-02. URL: <https://www.docker.com/products/docker-swarm> (cited on pages 652, 733).
- [172] *Microsoft Docs Azure Stream Analytics Documentation*. Web Page. Accessed: 2017-02-03. URL: <https://docs.microsoft.com/en-us/azure/stream-analytics/> (cited on page 683).
- [173] Jack Dongarra et al. “Accelerating numerical dense linear algebra calculations with GPUs”. In: *Numerical Computations with GPUs*. Springer, 2014, pages 3–28. URL: [http://link.springer.com/chapter/10.1007/978-3-319-06548-9\\_1](http://link.springer.com/chapter/10.1007/978-3-319-06548-9_1) (visited on 02/13/2017) (cited on page 655).
- [174] Dormando. *Memcached*. Web Page. accessed 2017-01-30. Feb. 2015. URL: <http://www.memcached.org/> (cited on page 695).
- [175] *Apache Drill*. Web Page. Accessed:2/4/2017. URL: <https://drill.apache.org/> (cited on page 677).
- [176] Wikipedia. *Dryad(Programming)-Wikipedia*. Web Page. Online,Accessed: 2017-02-24. Nov. 2016. URL: [https://en.wikipedia.org/wiki/Dryad\\_\(programming\)](https://en.wikipedia.org/wiki/Dryad_(programming)) (cited on page 647).
- [177] Michael Isard et al. “Dryad: distributed data-parallel programs from sequential building blocks”. In: *ACM SIGOPS operating systems review*. Volume 41. 3. Online,Accessed: 2017-02-24. ACM. 2007, pages 59–72. URL: <https://www.microsoft.com/en-us/research/wp-content/uploads/2007/03/eurosys07.pdf> (cited on page 647).
- [178] *Rebalancing in a multi-cloud environment in Science Cloud '13*. ACM, 2013, pages 21–28. ISBN: 978-1-4503-1979-9. DOI: 10.1145/2465848.2465854. URL: <http://dl.acm.org/citation.cfm?id=2465854> (cited on page 744).
- [179] Dylan Raithel. *Apache TinkerPop Graduates to Top-Level Project*. Web Page. 2016. URL: <https://www.infoq.com/news/2016/06/tinkerpop-top-level-apache/> (cited on page 661).
- [180] eclipse. *mqtt broker*. eclipse mosquito website. URL: <https://mosquitto.org/> (cited on page 440).
- [181] *EDUROAM*. Web Page. URL: <https://www.eduroam.org/about/> (cited on page 748).
- [182] UCAR Edward Hartnett and Rew RK. “Experience with an enhanced netCDF data model and interface for scientific data access”. In: *24th Conference on IIPS*. 2008 (cited on page 717).

- [183] *Ehcache - Features*. Web Page. Accessed: 2017-01-21. URL: <http://www.ehcache.org/about/features.html> (cited on page 696).
- [184] *Ehcache - Documentation*. Web Page. Accessed: 2017-01-21. URL: <http://www.ehcache.org/documentation/3.2/getting-started.html> (cited on page 696).
- [185] *Elastic Search*. Web Page. Accessed: 2017-1-25. URL: <https://www.elastic.co/products/elasticsearch> (cited on page 663).
- [186] *Elastic Search*. Web Page. Accessed: 2017-1-25. URL: <https://en.wikipedia.org/wiki/Elasticsearch> (cited on page 663).
- [187] *Elastic Search Getting Started*. Web Page. Accessed: 2017-1-25. URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html> (cited on page 663).
- [188] *Elastic Search on Hadoop*. Web Page. Accessed: 2017-1-25. URL: <https://www.elastic.co/products/hadoop> (cited on page 663).
- [189] elastic.io. *ELK stack*. elastic.io website. URL: <https://www.elastic.co/products> (cited on page 442).
- [190] *Elasticsearch*. en. Web Page. Page Version ID: 767434249. Feb. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Elasticsearch&oldid=767434249> (visited on 02/27/2017) (cited on page 663).
- [191] Elasticsearch. *Logstash Introduction*. Web Page. Accessed: 2017-02-11. Feb. 2017. URL: <https://www.elastic.co/guide/en/logstash/current/introduction.html> (cited on page 663).
- [192] Cliff Engle et al. “Shark: Fast Data Analysis Using Coarse-grained Distributed Memory”. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’12. Scottsdale, Arizona, USA: ACM, 2012, pages 689–692. ISBN: 978-1-4503-1247-9. DOI: 10.1145/2213836.2213934. URL: <http://doi.acm.org/10.1145/2213836.2213934> (cited on page 673).
- [193] Erlang Central. *Couchbase Performance and Scalability: Iterating with DTrace Observability*. Web page. Online; accessed 29-jan-2017. Mar. 2012. URL: <http://erlangcentral.org/videos/couchbase-performance-and-scalability-iterating-with-dtrace-observability/#.WI5uYephnRY> (cited on page 709).
- [194] erlang-mqtt. *erlang mqtt broker*. wmqtt website. URL: <http://emqtt.io/docs/v2/index.html> (cited on page 442).
- [195] Christian Esposito et al. “A Knowledge-based Platform for Big Data Analytics Based on Publish/Subscribe Services and Stream Processing”. In: *Know.-Based Syst.* 79.C (May 2015). Accessed: 2017-1-27, pages 3–17. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2014.05.003. URL: <http://dx.doi.org/10.1016/j.knosys.2014.05.003> (cited on page 688).
- [196] Patrick Th. Eugster et al. “The Many Faces of Publish/Subscribe”. In: *ACM Comput. Surv.* 35.2 (June 2003). Accessed: 2017-1-27, pages 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: <http://doi.acm.org/10.1145/857076.857078> (cited on page 688).
- [197] *Exploring Big Data with Helix: Finding Needles in a Big Haystack*. Web Page. URL: [https://sigmodrecord.org/publications/sigmodRecord/1412/pdfs/09\\_industry\\_Ellis.pdf](https://sigmodrecord.org/publications/sigmodRecord/1412/pdfs/09_industry_Ellis.pdf) (cited on page 721).
- [198] Facebook Inc. *Under the Hood: Scheduling MapReduce jobs more efficiently with Corona*. Web Page. accessed 2017-02-13. Nov. 2012. URL: <https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920/> (cited on page 722).



- [199] *Facebook's Graph Search puts Apache Giraph on the map*. web page. URL: <https://www.pcworld.com/article/2046680/facebooks-graph-search-puts-apache-giraph-on-the-map.html> (cited on page 686).
- [200] *Facebook's New Realtime Analytics System: HBase To Process 20 Billion Events Per Day*. Web Page. Accessed:2017-02-8. Mar. 2011. URL: <http://highscalability.com/blog/2011/3/22/facebooks-new-realtime-analytics-system-hbase-to-process-20.html> (cited on pages 682, 683).
- [201] smart factory. *MQTT and Kibana - Open source Graphs and Analysis for IoT*. smart factory website. May 2016. URL: <https://smart-factory.net/mqtt-and-kibana-open-source-graphs-and-analysis-for-iot/> (cited on page 442).
- [202] smart factory. *Storing IoT data using open source. MQTT and Elasticsearch - Tutorial*. smart factory website. Oct. 2016. URL: <https://smart-factory.net/mqtt-elasticsearch-setup/> (cited on page 442).
- [203] *Features*. Web Page. URL: <https://openvz.org/Features> (visited on 02/13/2017) (cited on page 742).
- [204] Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures". Doctoral Dissertation. University of California, Irvine, 2000. URL: [http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) (cited on page 756).
- [205] *FITS Vatican Library*. Web Page. URL: <https://www.vatlib.it/home.php?pag=digitalizzazione&ling=eng> (cited on page 717).
- [206] *Fits Matlab*. Web Page. URL: [https://www.mathworks.com/help/matlab/import\\_export/importing-flexible-image-transport-system-fits-files.html?requestedDomain=www.mathworks.com](https://www.mathworks.com/help/matlab/import_export/importing-flexible-image-transport-system-fits-files.html?requestedDomain=www.mathworks.com) (cited on page 717).
- [207] *FITS Nasa*. web. URL: <https://fits.gsfc.nasa.gov/> (cited on page 717).
- [208] *FITS News*. Web Page. URL: [https://fits.gsfc.nasa.gov/fits\\_standard.html](https://fits.gsfc.nasa.gov/fits_standard.html) (cited on page 717).
- [209] Apache Software Foundation. *About Apache Flex*. Web Page. Accessed: 2017-03-01. Feb. 2017. URL: <http://flex.apache.org/about-what-is.html> (cited on page 757).
- [210] Joab Jackson. "Adobe donates Flex to Apache". In: *The IDG News Service* (Nov. 2011). Accessed: 2011-11-17. URL: <https://www.techworld.com.au/article/407714/adobe-donates-flex-apache> (cited on page 757).
- [211] Licia Florio and Klaas Wierenga. "Eduroam, providing mobility for roaming users". In: *TERENA* (2005). URL: <https://www.terena.org/activities/tf-mobility/docs/ppt/eunis-eduroamfinal-LF.pdf> (cited on page 748).
- [212] Google. *Google App Engine*. webpage. Accessed : 04-09-2017. URL: <https://research.google.com/pubs/pub35650.html> (cited on page 648).
- [213] For Dummies, Inc. *Cloudera Imapala and Hadoop*. Web page. Online; accessed 7-Apr-2017. URL: <http://www.dummies.com/programming/big-data/hadoop/cloudera-impala-and-hadoop/> (cited on page 674).
- [214] Ian Foster. "Globus toolkit version 4: Software for service-oriented systems". In: *Journal of computer science and technology* 21.4 (2006), page 513 (cited on page 724).
- [215] *Foundation >> OpenStack Open Source Cloud Computing Software*. Web Page. Online; accessed 23-Feb-2017. URL: <https://www.openstack.org/foundation/> (cited on page 743).
- [216] Apache Software Foundation. *Apache Parquet*. Web Page. Accessed: 02-06-2017. URL: <https://parquet.apache.org/documentation/latest> (cited on page 718).
- [217] Apache Software Foundation. *Kafka-A distributed streaming platform*. Web Page. URL: <https://kafka.apache.org/> (cited on page 691).

- [218] Apache Software Foundation. *GenApp - Apache Airavata - Apache Software Foundation*. Web page. Accessed: 2017-02-22. Aug. 2014. URL: <https://cwiki.apache.org/confluence/display/AIRAVATA/GenApp> (cited on page 644).
- [219] Apache Software Foundation. *Apache Airavata*. Web page. Accessed: 2017-02-22. 2016. URL: <http://airavata.apache.org> (cited on page 644).
- [220] The Apache Software Foundation. *Trident Tutorial*. Web Page. Accessed: 2017-1-24. URL: <http://storm.apache.org/releases/0.10.1/Trident-tutorial.html> (cited on page 645).
- [221] The Eclipse Foundation. *Eclipse Winery*. Web Page. Accessed: 2017-1-24. URL: <https://projects.eclipse.org/projects/soa.winery> (cited on page 738).
- [222] Tony Fountain et al. “The Open Source DataTurbine Initiative: Empowering the Scientific Community with Streaming Data Middleware”. In: *Bulletin - Ecological Society of America* 93.3 (July 2012), pages 242–252. URL: <http://onlinelibrary.wiley.com/doi/10.1890/0012-9623-93.3.242/abstract> (cited on page 684).
- [223] G. Fox and S. Pallickara. “Deploying the NaradaBrokering Substrate in Aiding Efficient Web and Grid Service Interactions”. In: *Proceedings of the IEEE* 93.3 (Mar. 2005). Accessed : 2017-02-15, pages 564–577. ISSN: 0018-9219. DOI: [10.1109/JPROC.2004.842759](https://doi.org/10.1109/JPROC.2004.842759) (cited on page 691).
- [224] *FTP Wikipedia*. Web Page. Accessed: 2017-02-02. URL: <https://cwiki.apache.org/confluence/display/Hive/RCFileCat> (cited on page 719).
- [225] *FUSE Site*. Web Page. URL: <http://fuse.sourceforge.net> (cited on page 727).
- [226] Edgar Gabriel et al. “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. In: *Proceedings, 11th European PVM/MPI Users’ Group Meeting*. Budapest, Hungary, Sept. 2004, pages 97–104. URL: <https://www.open-mpi.org/papers/euro-pvmmpi-2004-overview/euro-pvmmpi-2004-overview.pdf> (cited on page 752).
- [227] *About Ansible Galaxy*. Web Page. Accessed: 2017-02-06. URL: <https://galaxy.ansible.com/intro/> (cited on page 646).
- [228] *GitHub ansible/ galaxy*. Web Page. Accessed: 2017-02-06. URL: <https://github.com/ansible/galaxy/> (cited on page 646).
- [229] Luke Galea. *Graylog2 optimization for High-log Environments*. webpage. Accessed : 02-21-2017. July 2012. URL: <https://dzone.com/articles/graylog2-optimization-high-log> (cited on page 664).
- [230] *Galera Cluster*. Web Page. 2017. URL: <http://galeracluster.com/> (cited on page 702).
- [231] Gamefromscratch. *Hands On With Amazon’s Lumberyard Game Engine - YouTube*. URL: <https://www.youtube.com/watch?v=FUD1TTbt4qE> (visited on 02/27/2017) (cited on page 679).
- [232] *Ganglia Monitoring System*. Web Page. Accessed: 2017-02-24. URL: <http://ganglia.info/> (cited on page 747).
- [233] *Ganglia (software)*. Web Page. Accessed: 2017-02-24. URL: [https://en.wikipedia.org/wiki/Ganglia\\_\(software\)](https://en.wikipedia.org/wiki/Ganglia_(software)) (cited on page 747).
- [234] Mitch Garnaat. *boto components*. Web Page. Accessed: 2017-1-25. URL: <http://boto.cloudhackers.com/en/latest/> (cited on page 734).
- [235] Mitch Garnaat. *boto3-documentation components*. Web Page. Accessed: 2017-1-26. URL: <https://boto3.readthedocs.io/en/latest/> (cited on page 734).
- [236] Mitch Garnaat. *boto-amazon-python-sdk components*. Web Page. Accessed: 2017-1-26. URL: <https://aws.amazon.com/sdk-for-python/> (cited on page 734).
- [237] Mitch Garnaat. *boto-github components*. Web Page. Accessed: 2017-1-25. URL: <https://github.com/boto/boto3> (cited on page 734).

- [238] Robert C. Gentleman et al. “Bioconductor: open software development for computational biology and bioinformatics”. In: *Genome Biology* 5.10 (2004), R80. ISSN: 1474-760X. DOI: [10.1186/gb-2004-5-10-r80](https://doi.org/10.1186/gb-2004-5-10-r80). URL: <http://dx.doi.org/10.1186/gb-2004-5-10-r80> (cited on page 654).
- [239] *Get started with Azure Blob storage (object storage) using .NET | Microsoft Docs*. Web Page. URL: <https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-blobs> (visited on 02/13/2017) (cited on page 728).
- [240] Ali Ghodsi et al. “Dominant Resource Fairness: Fair Allocation of Multiple Resource Types.” In: *NSDI*. Volume 11. 11. 2011, pages 24–24 (cited on page 721).
- [241] *GitHub - elastic/kibana: Kibana analytics and search dashboard for Elasticsearch*. Web Page. URL: <https://github.com/elastic/kibana> (visited on 02/27/2017) (cited on page 663).
- [242] *GitHub - jupyter/jupyter: Jupyter metapackage for installation, docs and chat*. Web page. URL: <https://github.com/jupyter/jupyter> (visited on 02/27/2017) (cited on page 646).
- [243] *GitHub - jupyter/notebook: Jupyter Interactive Notebook*. Web page. URL: <https://github.com/jupyter/notebook> (visited on 02/27/2017) (cited on page 646).
- [244] *CDAP Applications*. Code Repository. Accessed: 2017-02-18. May 2015. URL: <https://github.com/caskdata/cdap-apps> (cited on page 754).
- [245] *coreos/rkt*. Web Page. Accessed: 1/27/2017. URL: <https://github.com/coreos/rkt/> (cited on page 744).
- [246] *About the Globus Toolkit*. Web Page. Accessed: 2017-02-26. URL: <http://toolkit.globus.org/toolkit/about.html> (cited on page 724).
- [247] Gogle Inc. *Google Cloud Datastore Overview*. Web page. Online; accessed 9-Apr-2017. URL: <https://cloud.google.com/datastore/docs/concepts/overview> (cited on page 716).
- [248] Sunila Gollapudi. *Getting Started with Greenplum for Big Data Analytics*. Packt Publishing, Oct. 25, 2013. 172 pages. ISBN: 1782177043. URL: [http://www.ebook.de/de/product/21653990/sunila\\_gollapudi\\_getting\\_started\\_with\\_greenplum\\_for\\_big\\_data\\_analytics.html](http://www.ebook.de/de/product/21653990/sunila_gollapudi_getting_started_with_greenplum_for_big_data_analytics.html) (cited on page 720).
- [249] Jose Gonzalez and Jeff Lindsay. *Buildstep*. Code repository. Accessed: 2017-1-24. July 2015. URL: <https://github.com/progrium/buildstep> (cited on page 738).
- [250] Tom Goodale et al. *A Simple API for Grid Applications*. webpage. Accessed : 02-01-2017. Sept. 2013. URL: <https://www.ogf.org/documents/GFD.90.pdf> (cited on page 732).
- [251] Google. *Cloud Bigtable*. Web Page. accessed 2017-01-29. URL: <https://cloud.google.com/bigtable/> (cited on page 710).
- [252] Google. *CLOUD DATAFLOW*. WebPage. URL: <https://cloud.google.com/dataflow/> (cited on page 650).
- [253] Google. *Cloud Dataflow*. Web Page. Accessed: 02/03/2016. URL: <https://cloud.google.com/dataflow/> (cited on page 679).
- [254] Google. *Google Cloud Prediction API Documentation*. Web Page. Accessed 2017-1-26. URL: <https://cloud.google.com/prediction/docs/> (cited on page 656).
- [255] Google. *Google Cloud Translation API Documentation*. Web Page. Accessed 2017-02-20. URL: <https://cloud.google.com/translate/docs/> (cited on page 656).
- [256] Google. *Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud*. 2012. URL: <http://homes.cs.washington.edu/~alon/files/socc10.pdf> (cited on page 662).
- [257] Google. *FusionTableSupportHelp*. Web Page. 2017. URL: <https://support.google.com/fusiontables/answer/171181?hl=en> (cited on page 662).

- [258] Mike Burrows. *Chubby Site*. Web Page. URL: <https://research.google.com/archive/chubby.html> (cited on page 750).
- [259] *Google Cloud*. Webpage. URL: [https://en.wikipedia.org/wiki/Google\\_Cloud](https://en.wikipedia.org/wiki/Google_Cloud) (cited on page 746).
- [260] *Google cloud platform*. Webpage. URL: <https://cloud.google.com> (cited on page 746).
- [261] *Google Cloud SQL*. web page. URL: <https://cloud.google.com/sql/> (cited on page 703).
- [262] *Google Cloud SQL FAQ*. Webpage. URL: <https://cloud.google.com/sql/faq#version> (cited on page 704).
- [263] *Google Cloud Storage Documentation*. Web Page. Accessed: 02-06-2017. URL: <https://cloud.google.com/storage/docs> (cited on page 729).
- [264] Google Developers. *Cloud Machine Learning*. Web page. accessed 16-March-2017. URL: <https://cloud.google.com/ml-engine/> (cited on page 758).
- [265] Google Developers. *Cloud ML Engine Overview*. Web page. accessed 16-March-2017. URL: <https://cloud.google.com/ml-engine/docs/concepts/technical-overview> (cited on page 758).
- [266] Google Inc. *Balancing Strong and Eventual Consistency with Google Cloud Datastore*. Web page. Online; accessed 9-Apr-2017. URL: <https://cloud.google.com/datastore/docs/articles/balancing-strong-and-eventual-consistency-with-google-cloud-datastore/> (cited on page 716).
- [267] Google Inc. *Google-pub-sub-scalable-messaging-middleware*. Web Page. Accessed: 2017-02-10. URL: <https://cloud.google.com/pubsub/> (cited on page 694).
- [268] Google Inc. *What-is-Google-Pub-Sub*. Web Page. Accessed: 2017-02-09. URL: <https://cloud.google.com/pubsub/docs/overview> (cited on page 694).
- [269] *Gora - in-memory data model and persistence for big data*. Web Page. Accessed: 2017-01-18. URL: <http://gora.apache.org/> (cited on page 695).
- [270] Pivotal Software Inc. *gpfdist*. Web Page. 2017. URL: [http://gpdb.docs.pivotal.io/4330/utility\\_guide/admin\\_utilities/gpfdist.html](http://gpdb.docs.pivotal.io/4330/utility_guide/admin_utilities/gpfdist.html) (cited on page 720).
- [271] GraphLab. *GraphLab*. Web Page. accessed 2017-01-28. Dec. 2012. URL: <http://www.select.cs.cmu.edu/code/graphlab/> (cited on page 660).
- [272] *Dashboards - 2.2.1 documentation*. webpage. Accessed : 02-21-2017. 2012. URL: <http://docs.graylog.org/en/2.2/pages/dashboards.html> (cited on page 664).
- [273] Rick Grehan. *NoSQL Showdown: MongoDB vs. Couchbase*. Web page. Online; accessed 29-jan-2017. Mar. 2013. URL: <http://www.infoworld.com/article/2613970/nosql/nosql-showdown--mongodb-vs--couchbase.html> (cited on page 709).
- [274] *Globus Online (GridFTP)*. Web Page. Accessed:1/16/2017. URL: <https://en.wikipedia.org/wiki/GridFTP> (cited on page 719).
- [275] C.J. Gronlund et al. *Azure Machine Learning*. Web Page. Accessed: 2017-02-27. Jan. 2017. URL: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-what-is-machine-learning#what-is-machine-learning-in-the-microsoft-azure-cloud> (cited on page 656).
- [276] RDF Working Group. *Resource Description Framework (RDF)*. Online. Feb. 2014. URL: <https://www.w3.org/RDF/> (cited on page 712).
- [277] *Grow Hosting Business with Software Platform*. Web Page. URL: <https://jelastic.com/cloud-business-for-hosting-providers/> (visited on 02/13/2017) (cited on page 669).
- [278] *H2O Website Documentation components*. Web Page. URL: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html> (cited on page 659).



- [279] *H2O Wikipedia Documentation components*. Web Page. Accessed : 2017-02-12. URL: [https://en.wikipedia.org/wiki/H2O\\_\(software\)](https://en.wikipedia.org/wiki/H2O_(software))1 (cited on page 659).
- [280] Hadoop Apache. *Apache Software Foundation*. Web Page. 2016. URL: <https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html> (cited on page 721).
- [281] Bajda Pawlikowski et al. *An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads*. Web Page. Accessed: 2017-02-12. URL: [db.cs.yale.edu/hadoopdb/hadoopdb.html](http://db.cs.yale.edu/hadoopdb/hadoopdb.html) (cited on page 675).
- [282] Azzam Haidar, Jakub Kurzak, and Piotr Luszczek. “An improved parallel singular value algorithm and its implementation for multicore hardware”. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, page 90. URL: <http://dl.acm.org/citation.cfm?id=2503292> (visited on 02/13/2017) (cited on page 655).
- [283] H. F. Halaoui. “A spatio temporal indexing structure for efficient retrieval and manipulation of discretely changing spatial data”. In: *Journal of Spatial Science* 53.2 (2008), pages 1–12. DOI: 10.1080/14498596.2008.9635146. URL: <http://dx.doi.org/10.1080/14498596.2008.9635146> (cited on page 665).
- [284] C Hamer. *Setup Raspberry pi as a dhcp server*. Web Page. accessed: 2018-2-26. Sept. 2014. URL: <https://blog.monotok.org/setup-raspberry-pi-dhcp-server/> (cited on page 827).
- [285] Matthew Hardin. *Understanding LMDB Database File Sizes and Memory Utilization*. WebPage. May 2016. URL: <https://symas.com/understanding-lmdb-database-file-sizes-and-memory-utilization/> (cited on page 696).
- [286] Harp. *Harp*. Web Page. accessed 2017-01-28. Jan. 2012. URL: <http://harpjs.com> (cited on page 689).
- [287] *Haystack*. Web Page. URL: [www.project-haystack.org](http://www.project-haystack.org) (cited on page 725).
- [288] Hazelcast. *Open Source In-Memory Data Grid*. Code Repository. accessed 2017-01-29. Jan. 2017. URL: <https://github.com/hazelcast/hazelcast> (cited on page 696).
- [289] *hdfs*. Web Page. Accessed: 2017-1-21. URL: <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html> (cited on page 725).
- [290] Yongqiang He et al. “RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems”. In: *Data Engineering (ICDE), 2011 IEEE 27th International Conference on Engineering*. IEEE. 2011, pages 1199–1208 (cited on page 718).
- [291] Michael Heap. *Ansible From Beginner to Pro*. Edited by L. Corrigan et al. apress, 2016 (cited on page 646).
- [292] *Heat - Openstack*. webpage. Accessed : 01-15-2017. URL: <https://wiki.openstack.org/wiki/Heat> (cited on page 735).
- [293] *Heroku*. Web Page. Accessed:1/27/2017. URL: <https://devcenter.heroku.com/> (cited on page 667).
- [294] Hugo Hiden et al. *e-Science Central: Cloud-based e-Science and its application to chemical property modelling*. Technical report. Newcastle, England: University of Newcastle upon Tyne, Nov. 2010. URL: [http://eprint.ncl.ac.uk/file\\_store/production/179301/8EA32A5C-DE97-44A8-869B-81731044F2A8.pdf](http://eprint.ncl.ac.uk/file_store/production/179301/8EA32A5C-DE97-44A8-869B-81731044F2A8.pdf) (cited on page 650).
- [295] *hivemq.intrwebsite mqtt*. hivemq website. URL: <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt> (cited on page 439).
- [296] *Home - Microsoft/CNTK Wiki - GitHub*. Web Page. URL: <https://github.com/Microsoft/CNTK/wiki> (visited on 02/13/2017) (cited on page 666).

- [297] *How Puppet Works*. Web Page. URL: <http://www.slashroot.in/puppet-tutorial-how-does-puppet-work> (cited on page 733).
- [298] *How To Create OpenVZ Container In OpenVZ | Unixmen*. Web Page. URL: <https://www.unixmen.com/how-to-create-openvz-container-in-openvz/> (visited on 02/13/2017) (cited on page 742).
- [299] *How To Use Logstash and Kibana To Centralize Logs On Ubuntu 14.04*. Web Page. URL: <https://www.digitalocean.com/community/tutorials/how-to-use-logstash-and-kibana-to-centralize-and-visualize-logs-on-ubuntu-14-04> (visited on 02/27/2017) (cited on page 663).
- [300] Philip Howard. “Blazegraph GPU”. In: White Paper. Dec. 2015, page 13. URL: [https://www.blazegraph.com/whitepapers/Blazegraph-gpu\\_InDetail\\_BloorResearch.pdf](https://www.blazegraph.com/whitepapers/Blazegraph-gpu_InDetail_BloorResearch.pdf) (cited on page 714).
- [301] HowStuffWorks.com. *What are relational databases?* Online. Mar. 2001. URL: <http://computer.howstuffworks.com/question599.htm> (cited on page 701).
- [302] Robert Kallman et al. “H-Store: a High-Performance, Distributed Main Memory Transaction Processing System”. In: *Proc. VLDB Endow.* 1.2 (2008), pages 1496–1499. ISSN: 2150-8097. DOI: <http://doi.acm.org/10.1145/1454159.1454211>. URL: <http://hstore.cs.brown.edu/papers/hstore-demo.pdf> (cited on page 697).
- [303] HPE. *HPE Helion Stackato*. Webpage. URL: <https://www.hpe.com/us/en/software/multi-cloud-platform.html> (cited on page 669).
- [304] *H-Store*. Web Page. Accessed: 1/16/2017. URL: <http://hstore.cs.brown.edu/> (cited on page 697).
- [305] *H-Store Wiki*. Web Page. Accessed: 1/16/2017. URL: <https://en.wikipedia.org/wiki/H-Store> (cited on page 697).
- [306] IBM. *BlueworksLive*. Web Page. Accessed: 2017-1-24. URL: <https://www.blueworkslive.com/home> (cited on page 739).
- [307] IBM. *Exercise: Analyze business processes with IBM BPM Blueprint*. Web Page. Accessed: 2017-1-24. URL: <http://www.ibm.com/developerworks/downloads/soasandbox/blueprint.html> (cited on page 739).
- [308] IBM. *IBM Blueworks Live*. Web Page. Accessed: 2017-1-24. URL: [https://en.wikipedia.org/wiki/IBM\\_Blueworks\\_Live](https://en.wikipedia.org/wiki/IBM_Blueworks_Live) (cited on page 739).
- [309] IBM. *IBM Spectrum Scale*. Web Page. accessed 2017-01-29. URL: <http://www-03.ibm.com/systems/storage/spectrum/scale/> (cited on page 727).
- [310] IBM. *What is Flume?* web page. accessed 201-02-06. URL: <https://www-01.ibm.com/software/data/infosphere/hadoop/flume/> (cited on page 720).
- [311] IBM Corporation. *IBM System G documentation*. Web Page. Accessed: 2017-02-19. IBM, 2014. URL: <http://systemg.research.ibm.com/> (cited on page 660).
- [312] IBM Corporation. *IBM System G documentation-2*. Web Page. Accessed: 2017-02-17. Predictive Analytics Today, 2017. URL: <http://www.predictiveanalyticstoday.com/ibm-system-g-native-store/> (cited on page 660).
- [313] ibm. *Extreme cloud administration toolkit*. Webpage. URL: <http://www-03.ibm.com/systems/technicalcomputing/xcat/> (cited on page 735).
- [314] IBM Corp. Web Page. Online; accessed 25-jan-2017. Jan. 2017. URL: [www.softlayer.com](http://www.softlayer.com) (cited on page 668).
- [315] IBM Corp. Web Page. Online; accessed 25-jan-2017. Jan. 2017. URL: <https://www.ibm.com/cloud-computing/bluemix/> (cited on page 668).
- [316] *IBM dashDB*. Web Page. URL: <https://www.ibm.com/analytics/us/en/technology/cloud-data-services/dashdb/> (cited on page 705).

- [317] *IBM Watson*. Web Page. Accessed: 2017-1-25. URL: [https://en.wikipedia.org/wiki/Watson\\_\(computer\)](https://en.wikipedia.org/wiki/Watson_(computer)) (cited on page 659).
- [318] *IBM Watson Product Page*. Web Page. Accessed: 2017-1-25. URL: <https://www.ibm.com/watson> (cited on page 659).
- [319] Roger Ignazio. *Mesos in Action*. 1st. Greenwich, CT, USA: Manning Publications Co., May 17, 2016. 272 pages. ISBN: 1617292923. URL: <http://dl.acm.org/citation.cfm?id=3006364> (cited on page 721).
- [320] ImageJ. *ImageJ Introduction*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: <https://imagej.nih.gov/ij/docs/intro.html> (cited on page 654).
- [321] *Inca - Periodic, automated, user-level cyberinfrastructure testing*. Web Page. Accessed: 2017-01-16. URL: <http://inca.sdsc.edu/> (cited on page 747).
- [322] InCommon. *What is the InCommon Federation?* Webpage. URL: [https://spaces.internet2.edu/download/attachments/2764/final\\_InCommon.pdf](https://spaces.internet2.edu/download/attachments/2764/final_InCommon.pdf) (cited on page 748).
- [323] Indiana University et al. *SEAGrid Portal*. Web page. Accessed: 2017-02-22. July 2016. URL: <https://seagrid.org> (cited on page 644).
- [324] Dexter Industries. *Grovepi*. Dexter Industries website. 2017. URL: <https://www.dexterindustries.com/grovepi/> (cited on page 444).
- [325] *Infinispan*. Webpage. URL: <https://en.wikipedia.org/wiki/Infinispan> (cited on page 697).
- [326] *Intel Data Analytics Acceleration Library (Intel DAAL)*. Web Page. Accessed: 02-23-2017. URL: <https://software.intel.com/en-us/intel-daal> (cited on page 658).
- [327] *Intel Graph Analytics Solutions. Intel Graph Builder for Apache Hadoop\* Software v2*. Web Page. URL: <https://www-ssl.intel.com/content/www/us/en/software/intel-graph-builder-for-apache-hadoop-software-v2-product-detail.html> (cited on page 661).
- [328] *Introducing Espresso - LinkedIn's hot new distributed document store | LinkedIn Engineerin*. Web Page. Online; accessed 23-Feb-2017. URL: <https://engineering.linkedin.com/espresso/introducing-espresso-linkedins-hot-new-distributed-document-store> (cited on page 708).
- [329] *Introduction | Kibana User Guide [5.2] | Elastic*. Web Page. Docs/Kibana/Reference/5.2. URL: <https://www.elastic.co/guide/en/kibana/current/introduction.html> (visited on 02/27/2017) (cited on page 663).
- [330] *Introduction to Infinispan. Distributed in-memory key/value data grid and cache*. Webpage. URL: <http://infinispan.org/about/> (cited on page 697).
- [331] *IPython*. en. Web page. Page Version ID: 767266837. Feb. 2017. URL: <https://en.wikipedia.org/w/index.php?title=IPython&oldid=767266837> (visited on 02/27/2017) (cited on page 646).
- [332] Muhammad Hussain Iqbal and Tariq Rahim Soomro. "Big Data Analysis: Apache Storm Perspective". In: *International Journal of Computer Trends and Technology (IJCTT)-2015*. Jan. 2015 (cited on page 680).
- [333] Romin Irani. *Docker Tutorial Series-Part 4-Docker Hub*. Blog. Accessed: 03-27-2018. July 2015. URL: <https://rominirani.com/docker-tutorial-series-part-4-docker-hub-b51fb545dd8e> (cited on page 500).
- [334] iRod. Web Page. URL: <https://irods.org/> (cited on page 716).
- [335] iRod. Web Page. URL: <https://github.com/irods/irods> (cited on page 716).
- [336] Mohammad Islam et al. "Oozie: towards a scalable workflow management system for hadoop". In: *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*. ACM. 2012, page 4 (cited on pages 647, 648).



- [337] C. Issariyapat et al. “Using Nagios as a groundwork for developing a better network monitoring system”. In: *2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies*. July 2012, pages 2771–2777 (cited on page 747).
- [338] *IU 'Twister' software improves Google's MapReduce for large-scale scientific data analysis*. Web Page. URL: <https://phys.org/news/2010-03-iu-twister-software-google-mapreduce.html> (visited on 02/13/2017) (cited on page 685).
- [339] *IU Twister software improves Google's MapReduce for large-scale scientific data analysis: IU News Room: Indiana University*. Web Page. URL: <http://newsinfo.iu.edu/news-archive/13726.html> (visited on 02/13/2017) (cited on page 685).
- [340] Jacob Jackson. *Google service analyzes live streaming data*. WebPage. June 2014. URL: <http://www.infoworld.com/article/2607938/data-mining/google-service-analyzes-live-streaming-data.html> (cited on page 650).
- [341] Jake Luciani. *Solandra Wiki*. Code Repository. Accessed: 2017-02-12. Feb. 2017. URL: <https://github.com/tjake/Solandra/wiki/Solandra-Wiki> (cited on page 706).
- [342] Jake Luciani. *Solandra Wiki*. Code Repository. Accessed: 2017-02-12. Feb. 2017. URL: <https://github.com/tjake/Solandra> (cited on page 706).
- [343] *Java Database Connectivity*. Web Page. Accessed: 2017-1-30. URL: [https://en.wikipedia.org/wiki/Java\\_Database\\_Connectivity](https://en.wikipedia.org/wiki/Java_Database_Connectivity) (cited on page 699).
- [344] *JClouds - The Java Multi-Cloud Toolkit*. Web Page. Accessed: 2017-01-20. URL: <https://jclouds.apache.org/> (cited on page 730).
- [345] *Jelastic*. en. Web Page. Page Version ID: 754931676. Dec. 2016. URL: <https://en.wikipedia.org/w/index.php?title=Jelastic&oldid=754931676> (visited on 02/13/2017) (cited on page 669).
- [346] Shantenu Jha et al. “Grid Interoperability at the Application Level Using SAGA”. In: *07 Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*. Dec. 2007. ISBN: 0769530648 (cited on page 732).
- [347] Ana Lucia Varbanescu Jianbin Fang and Henk Sips. “Sesame: A User-Transparent Optimizing Framework for Many-Core Processors”. In: *IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. 2013, pages 70–73 (cited on page 715).
- [348] Jitterbit. *Data Integration and ETL | Jitterbit | Integrate data from any source*. Web Page. accessed: 2017-1-26. Jitterbit. URL: <https://www.jitterbit.com/etl-data-integration/> (cited on page 651).
- [349] *Integration Made Easy - Jitterbit*. Accessed: 2017-1-26. Jitterbit,Inc. 1 Kaiser Plaza Suite 701 Oakland, CA 94612. URL: <http://www.jitterbit.com/Files/Product/Jitterbit-General-Datasheet.pdf> (cited on page 651).
- [350] *Technical Overview - Jitterbit*. Accessed: 2017-1-26. Jitterbit,Inc. 2011. URL: <http://www.jitterbit.com/Files/Product/JitterbitTechnicalOverview.pdf> (cited on page 651).
- [351] *The Java EE 6 Tutorial*. webpage. Accessed : 01-18-2017. URL: <http://docs.oracle.com/javaee/6/tutorial/doc/bnceh.html> (cited on page 692).
- [352] Tim Jones. *Anatomy of the libvirt virtualization*. Webpage. Jan. 2010. URL: <https://www.ibm.com/developerworks/library/l-libvirt/> (cited on page 729).
- [353] David Josephsen. *Nagios: Building Enterprise-Grade Monitoring Infrastructures for Systems and Networks*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2013. ISBN: 013313573X, 9780133135732 (cited on page 747).
- [354] The Apache Software Foundation. *Apache OpenJPA - Wikipedia*. webpage. Accessed : 02-22-2017. Jan. 2017. URL: [https://en.wikipedia.org/wiki/Apache\\_OpenJPA](https://en.wikipedia.org/wiki/Apache_OpenJPA) (cited on page 698).

- [355] JPAB. *JPA Performance Benchmark*. Web Page. Accessed: 2017-02-04. URL: <http://www.jpab.org/DataNucleus.html> (cited on page 699).
- [356] Canonical Ltd. *Juju*. Web Page. URL: <https://www.ubuntu.com/cloud/juju> (cited on page 735).
- [357] Guy E. Blelloch Julian Shun and Laxman Dhulipala. “Smaller and Faster: Parallel Processing of Compressed Graphs with Ligra+”. In: *ACM SIGPLAN Notices - PPOPP '13DCC '15 Proceedings of the 2015 Data Compression Conference*. Pittsburgh, Pennsylvania USA: IEEE Computer Society Washington, DC, USA, 2015, pages 403–412. ISBN: 978-1-4799-8430-55. DOI: 10.1109/DCC.2015.8. URL: <http://dl.acm.org/citation.cfm?id=2860198> (cited on page 687).
- [358] Steven van Wel Jurg Vliet Flavia Paganelli and Dara Dowd. *Elastic Beanstalk*. 1st edition. O'Reilly Media, Inc., Aug. 6, 2011. ISBN: 1449306640 (cited on page 668).
- [359] *Astronomical Image Processing with Hadoop*. Volume 442. Astronomical Data Analysis Software and Systems XX. ASP Conference Proceedings, July 2011. URL: <http://adsabs.harvard.edu/abs/2011ASPC..442...93W> (cited on page 718).
- [360] Riyad Kalla. *Well put! When should you use MongoDB vs Couchbase versus Redis...* Web page. Online; accessed 29-jan-2017. Oct. 2011. URL: <http://rick-hightower.blogspot.com/2014/04/well-put-when-should-you-use-mongodb-vs.html> (cited on page 709).
- [361] Mike Kavis. *THE PROS AND CONS OF PRIVATE AND PUBLIC PAAS*. Webpage. June 2013. URL: <https://www.virtualizationpractice.com/the-pros-and-cons-of-private-and-public-paas-21961/> (cited on page 669).
- [362] Keith-Bisset. *CINET - A Cyber-Infrastructure for Network Science*. Web Page. URL: [www.portal.futuresystems.org/project/233](http://www.portal.futuresystems.org/project/233) (cited on page 662).
- [363] Amol Kekre. *Apache Apex blog*. Web Page. Accessed: 2017-02-26. Sept. 2015. URL: <https://www.datatorrent.com/blog/introducing-apache-apex-incubating/> (cited on page 756).
- [364] Ben Kepes. *Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS*. white paper. Rackspace, 2015. URL: <https://support.rackspace.com/white-paper/understanding-the-cloud-computing-stack-saas-paas-iaas/> (cited on page 670).
- [365] Kepler Project. *The Kepler Project*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: <https://kepler-project.org> (cited on page 644).
- [366] Justin Kestelyn. *Phoenix in 15 Minutes or Less*. Web page. Online; accessed 25-jan-2017. Mar. 2013. URL: <http://blog.cloudera.com/blog/2013/03/phoenix-in-15-minutes-or-less/> (cited on page 674).
- [367] Agargenta. *About kestrel*. Web Page. Accessed: 2017-02-12. URL: <https://github.com/twitter-archive/kestrel> (cited on page 692).
- [368] *Keystone Wiki*. Web Page. Accessed: 2017-02-12. URL: <https://wiki.openstack.org/wiki/Keystone> (cited on page 748).
- [369] *Kibana*. en. Web Page. Page Version ID: 767434139. Feb. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Kibana&oldid=767434139> (visited on 02/27/2017) (cited on page 663).
- [370] *Kibana: Explore, Visualize, Discover Data |Elastic*. Web Page. URL: <https://www.elastic.co/products/kibana> (visited on 02/27/2017) (cited on page 663).
- [371] Jik-Soo Kim et al. “HTCaaS: Leveraging Distributed Supercomputing Infrastructures for Large-Scale Scientific Computing”. In: *ACM MTAGS 13*. 2013 (cited on page 725).
- [372] *Kinesis - real-time streaming data in the AWS cloud*. Web Page. Accessed: 2017-01-17. URL: <https://aws.amazon.com/kinesis/> (cited on page 681).

- [373] Wikipedia. *Apache OpenJPA*. webpage. Accessed : 02-22-2017. Dec. 2016. URL: <http://openjpa.apache.org/> (cited on page 698).
- [374] Oliver Kopp et al. “Winery - A Modeling Tool for TOSCA-based Cloud Applications”. In: *11<sup>th</sup> International Conference on Service-Oriented Computing*. Springer, Dec. 2013, pages 700–704. URL: [http://www.iaas.uni-stuttgart.de/RUS-data/INPROC-2013-46%5C%20-%5C%20Winery\\_A\\_Modeling\\_Tool\\_for\\_TOSCA-based\\_Cloud\\_Applications.pdf](http://www.iaas.uni-stuttgart.de/RUS-data/INPROC-2013-46%5C%20-%5C%20Winery_A_Modeling_Tool_for_TOSCA-based_Cloud_Applications.pdf) (cited on page 738).
- [375] Lars Kotthoff. *LLAMA: Leveraging Learning to Automatically Manage Algorithms*. Research paper. Apr. 30, 2014. URL: <https://arxiv.org/abs/1306.1031> (cited on page 722).
- [376] T. Kraska et al. “MLbase: A Distributed Machine Learning System”. In: *MLbase: A Distributed Machine Learning System*. Conference on Innovative Data Systems Research. 2013 (cited on page 653).
- [377] Kubernetes. *Kubernetes Site*. Web Page. URL: <https://kubernetes.io/docs/whatisk8s/> (cited on page 737).
- [378] Jakub Kurzak et al. “Scheduling dense linear algebra operations on multicore processors”. In: *Concurrency and Computation: Practice and Experience* 22.1 (2010), pages 15–44. URL: <http://onlinelibrary.wiley.com/doi/10.1002/cpe.1467/full> (visited on 02/13/2017) (cited on page 655).
- [379] *KVM Wikipedia Documentation*. Web Page. Accessed : 2017-02-12. URL: [https://en.wikipedia.org/wiki/Kernel-based\\_Virtual\\_Machine](https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine) (cited on page 741).
- [380] *KVM webpage documentation*. Web Page. Accessed : 2017-02-13. URL: [http://www.linux-kvm.org/page/Main\\_Page](http://www.linux-kvm.org/page/Main_Page) (cited on page 741).
- [381] *Kyoto Cabinet*. Web Page. Accessed:1/16/2017. URL: <http://fallabs.com/kyotocabinet/> (cited on page 678).
- [382] *Kyoto Cabinet*. Web Page. Accessed: 2017-1-27. URL: <http://fallabs.com/kyotocabinet/> (cited on page 707).
- [383] Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. “GraphChi: Large-scale Graph Computation on Just a PC”. In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*. USENIX Association, 2012, pages 31–46. ISBN: 978-1-931971-96-6. URL: <http://dl.acm.org/citation.cfm?id=2387880.2387884> (cited on page 687).
- [384] Fal labs. *Nijo Cabinet: a straightforward implementation of DBM*. Web Page. Accessed: 02/03/2016. URL: <http://fallabs.com/kyotocabinet/> (cited on page 708).
- [385] Fal labs. *Shijo Cabinet: a handy cache/storage server*. Web Page. Accessed: 02/03/2016. URL: <http://fallabs.com/kyototycoon/> (cited on page 707).
- [386] Gregor von Laszewski and Mike Hategan. “Using Karajan”. In: *Java CoG Kit Karajan/ Gridant Workflow Guide*. 2005. URL: <https://pdfs.semanticscholar.org/48e1/31ca4e7a76ca98c43d46975cd79c3dbfd4cb.pdf> (cited on page 758).
- [387] Joe Lennon. *Exploring CouchDB: A document-oriented database for Web applications*. Web page. accessed feb-26-2017. Mar. 2009. URL: <http://www.ibm.com/developerworks/opensource/library/os-couchdb/index.html> (cited on page 709).
- [388] LevelDB. *LevelDB*. Web Page. accessed 2017-02-25. Feb. 2017. URL: [leveldb.org](http://leveldb.org) (cited on page 711).
- [389] T. Li et al. “ZHT: A Light-Weight Reliable Persistent Dynamic Scalable Zero-Hop Distributed Hash Table”. In: *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. May 2013, pages 775–787. DOI: 10.1109/IPDPS.2013.110. URL: [http://datasys.cs.iit.edu/publications/2013\\_IPDPS13\\_ZHT.pdf](http://datasys.cs.iit.edu/publications/2013_IPDPS13_ZHT.pdf) (cited on page 707).

- [390] Weizhong Li. “Introduction to bioActors”. In: *bioKepler Tools and Its Applications*. Edited by bioKepler. 1st Workshop on bioKepler Tools and Its Applications 1. UCSD;SDSC. San Diego Supercomputer Center 10100 Hopkins Dr, La Jolla, CA 92093: bioKepler, Sept. 2012, page 31. URL: <http://www.biokepler.org/sites/swat.sdsc.edu.biokepler/files/workshops/2012-09-05/slides/2012-09-05-02-Li.pdf> (cited on page 646).
- [391] *Libcloud documentation*. Web Page. URL: <https://libcloud.readthedocs.io/en/latest/index.html> (cited on page 730).
- [392] *Libcloud wiki*. Web Page. URL: <https://wiki.apache.org/incubator/LibcloudProposal> (cited on page 730).
- [393] *Libvirt virtualization API*. Web Page. URL: <https://libvirt.org/> (cited on page 729).
- [394] Ching Yung Lin. “Graph Computing and Linked Big Data”. In: *8th IEEE/ICSC International Conference on Semantic Computing*. IBM Corporation, 2014, pages 1–63. URL: [http://iee-icsc.org/icsc2014/GraphComputing\\_IBM\\_Lin.pdf](http://iee-icsc.org/icsc2014/GraphComputing_IBM_Lin.pdf) (cited on page 661).
- [395] LinkedIn Developers. *Getting started with the REST API*. Web page. accessed 17-March-2017. URL: <https://developer.linkedin.com/docs/rest-api> (cited on page 682).
- [396] *Linux containers*. web page. URL: <https://en.wikipedia.org/wiki/LXC> (cited on page 742).
- [397] *Linux containers in use*. web page. URL: <http://www.infoworld.com/article/3072929/linux/containers-101-linux-containers-and-docker-explained.html> (cited on page 742).
- [398] Martin Lisa et al. *Data mining with iPlant*: Paper. Oct. 2014. URL: <https://academic.oup.com/jxb/article/66/1/1/2893405/Data-mining-with-iPlantA-meeting-report-from-the> (cited on page 672).
- [399] Scott Lowe. *An introduction to Openstack Heat*. webpage. Accessed : 01-15-2017. May 2014. URL: <http://blog.scottlowe.org/2014/05/01/an-introduction-to-openstack-heat/> (cited on page 735).
- [400] Andy Lurie. *Top 47 Log Management Tools*. webpage. Accessed : 02-21-2017. May 2014. URL: <http://blog.profitbricks.com/top-47-log-management-tools> (cited on page 664).
- [401] *LXD an hypervisor for containers*. web page. URL: <https://lists.linuxcontainers.org/pipermail/lxc-devel/2014-November/010817.html> (cited on page 755).
- [402] Tiago Macedo and Fred Oliveira. *Reddis Cookbook: Practical Technique for fast Data Manipulation*. Sonoma County, California, United States: O’Reilly Media, Aug. 2011. ISBN: 978-1-4493-0503-1 (cited on page 695).
- [403] *Apache Mahout*. Web Page. URL: <http://mahout.apache.org/> (cited on page 653).
- [404] Matthias Marschall. *Chef Infrastructure Automation Cookbook*. Packt Publishing, 2013. ISBN: 9351105164 and 9789351105169 (cited on page 733).
- [405] O. Martinez-Rubi et al. *Taming the beast: Free and open-source massive point cloud web visualization*. 2015. URL: <http://repository.tudelft.nl/islandora/object/uuid:0472e0d1-ec75-465a-840e-fd53d427c177?collection=research> (cited on page 665).
- [406] Karl Matthias and Sean P. Kane. *Docker Up and Running, Shipping Reliable Containers In Production*. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O’REILLY, 2015. ISBN: 978-1-491-91757-2 (cited on pages 652, 733).
- [407] Morel Matthieu. *S4 0.5.0 Overview*. Web Page. Accessed: 2017-02-24. Feb. 2013. URL: <https://cwiki.apache.org/confluence/display/S4/S4+0.5.0+Overview> (cited on page 680).



- [408] Norman Maurer and Marvin Wolfthal. *Netty in Action*. 1st. Greenwich, CT, USA: Manning Publications, Dec. 23, 2015. 296 pages. ISBN: 1617291471. URL: [http://www.ebook.de/de/product/21687528/norman\\_maurer\\_netty\\_in\\_action.html](http://www.ebook.de/de/product/21687528/norman_maurer_netty_in_action.html) (cited on page 689).
- [409] M. McLennan and R. Kennell. “HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering”. In: *Computing in Science Engineering* 12.2 (Mar. 2010), pages 48–53. ISSN: 1521-9615. DOI: [10.1109/mcse.2010.41](https://doi.org/10.1109/mcse.2010.41) (cited on page 671).
- [410] Eileen McNulty. *What Is Google Cloud Dataflow?* Web Page. Accessed: 02/03/2016. URL: <http://dataconomy.com/2014/08/google-cloud-dataflow/> (cited on page 679).
- [411] *Marionette Collective Webpage Documentation*. Web Page. Accessed: 2017-02-27. 2016. URL: <https://docs.puppet.com/mcollective/> (cited on page 693).
- [412] Sergey Melnik et al. “Dremel: Interactive Analysis of Web-Scale Datasets”. In: *Communications of the ACM* 54 (June 2011), pages 114–123. ISSN: 0001-0782. DOI: [10.1145/1953122.1953148](https://doi.org/10.1145/1953122.1953148). URL: <http://cacm.acm.org/magazines/2011/6/108648-dremel-interactive-analysis-of-web-scale-datasets/fulltext> (cited on page 676).
- [413] Xiangrui Meng et al. “MLlib: Machine Learning in Apache Spark”. In: *CoRR* abs/1505.06807 (2015). URL: <http://arxiv.org/abs/1505.06807> (cited on page 653).
- [414] Microsoft. *Event Hubs*. Web Page. accessed 2017-02-25. Dec. 2016. URL: <https://azure.microsoft.com/en-us/services/event-hubs/> (cited on page 694).
- [415] Microsoft Corp. *Form 10-K*. Web page. Online; accessed 21-jan-2017. July 2016. URL: <https://www.sec.gov/Archives/edgar/data/789019/000119312516662209/d187868d10k.htm> (cited on page 668).
- [416] Microsoft Corp. Web Page. Online; accessed 21-jan-2017. Jan. 2017. URL: <https://azure.microsoft.com/en-us/> (cited on page 668).
- [417] Microsoft Corp. *Designing a Scalable Partitioning Strategy for Azure Table Storage*. Web page. Online; accessed 28-jan-2017. Jan. 2017. URL: <https://docs.microsoft.com/en-us/rest/api/storageservices/fileservices/designing-a-scalable-partitioning-strategy-for-azure-table-storage> (cited on page 715).
- [418] Stuart Miniman. *Cisco Moves Up the Cloud Stack with Intelligent Automation*. webpage. 2011. URL: [http://wikibon.org/wiki/v/Cisco\\_Moves\\_Up\\_the\\_Cloud\\_Stack\\_with\\_Intelligent\\_Automation](http://wikibon.org/wiki/v/Cisco_Moves_Up_the_Cloud_Stack_with_Intelligent_Automation) (cited on page 736).
- [419] Ross Mistry and Stacia Misner. *Introducing Microsoft SQL Server 2014 Technical Overview*. Microsoft Press, 2014. ISBN: 978-0-7356-8475-1 (cited on page 700).
- [420] *Apache Spark’s MLlib*. Web Page. URL: <http://spark.apache.org/mllib/> (cited on page 653).
- [421] *MLPY documentation*. Web Page. 2012. URL: <http://mlpy.sourceforge.net/docs/3.5/> (cited on page 657).
- [422] AdaptiveComputing. *Moab Cluster Suite*. Web Page. Accessed: 2017-02-24. URL: <https://www.adaptivecomputing.com/products/> (cited on page 724).
- [423] Modine Austin. *Cobbler*. Web Page. accessed 2017-01-30. Feb. 2008. URL: [http://www.theregister.co.uk/2008/06/19/red\\_hat\\_summit\\_2008\\_cobbler/](http://www.theregister.co.uk/2008/06/19/red_hat_summit_2008_cobbler/) (cited on page 734).
- [424] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. “IaaS Cloud Architecture From Virtualized Datacenters to Federated Cloud Infrastructures”. In: *Computer*. Edited by IEEE. Volume 45. 12. IEEE Computer Society. IEEE, 2012, pages 65–72. DOI: [10.1109/MC.2012.76](https://doi.org/10.1109/MC.2012.76). URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6165242&isnumber=6383143> (cited on page 743).

- [425] Hive MQ. *MQTT Essentials Part 6: Quality of Service 0, 1 & 2*. Hivemq website. URL: <https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels> (cited on page 441).
- [426] Hive Mq. *MQTT Security Fundamentals: TLS / SSL*. hive mq website. URL: <https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl> (cited on pages 441, 442).
- [427] Hive mq. *MQTT Essentials Part 2: Publish & Subscribe*. HiveMQ website. URL: <https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe> (cited on pages 440, 441).
- [428] hive mq. *MQTT Security Fundamentals: OAuth 2.0 & MQTT*. hivemq website. URL: <https://www.hivemq.com/blog/mqtt-security-fundamentals-oauth-2-0-mqtt> (cited on page 442).
- [429] *MQTT*. Web Page. Accessed: 2017-1-27. URL: <http://mqtt.org/> (cited on page 693).
- [430] Mqtt. *Mqtt official website*. mqtt official website. URL: <http://mqtt.org/> (cited on page 440).
- [431] *MQTT Floodnet*. Web Page. Accessed: 2017-1-27. URL: <http://mqtt.org/projects/floodnet> (cited on page 693).
- [432] *MR-MPI*. Web Page. 2017. URL: <http://mapreduce.sandia.gov/doc> (cited on page 685).
- [433] Subramanian Muralidhar et al. “f4: Facebook’s warm blob storage system”. In: *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*. USENIX Association. 2014, pages 383–398 (cited on page 726).
- [434] Derek G. Murray et al. “Naiad: A Timely Dataflow System”. In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. SOSP ’13. Accessed: 2017-1-27. Farminton, Pennsylvania: ACM, 2013, pages 439–455. ISBN: 978-1-4503-2388-8. DOI: [10.1145/2517349.2522738](https://doi.org/10.1145/2517349.2522738). URL: <http://doi.acm.org/10.1145/2517349.2522738> (cited on page 647).
- [435] Colton Myers. *Learning SaltStack*. 2nd. Packt Publishing, 2015. ISBN: 978-1-78439-460-8, 1784394602. URL: <https://books.google.com/books?id=pVnsrQEACAAJ> (cited on page 734).
- [436] mysql. *MySQL 5.7 Reference Manual, What is MySQL*. Online. URL: <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html> (cited on page 701).
- [437] NASA/GSFC. *CDF User’s Guide (V3.3.6)*. Version 3.3.6. Accessed: 2017-1-28. NASA/GSFC Space Physics Data Facility. NASA / Goddard Space Flight Center, Greenbelt, Maryland 20771 (U.S.A.), Oct. 2016. URL: <http://spdf.gsfc.nasa.gov/pub/software/cdf/doc/cdf363/cdf363ug.pdf> (cited on page 717).
- [438] NASA/GSFC. *CDF Home Page*. webpage. accessed: 2017-1-28. NASA, Goddard Space Flight Center, 2017. URL: <http://cdf.gsfc.nasa.gov/> (cited on page 717).
- [439] National Instruments. *A Layered Approach to Designing Robot Software*. Web page. accessed 18-mar-2017. Mar. 2017. URL: <http://www.ni.com/white-paper/13929/en/> (cited on page 757).
- [440] Jeremy Nelson. *Mastering Redis*. Packt Publishing, May 2016. ISBN: 978-1783988181 (cited on page 696).
- [441] Neo4j. *Causal Cluster*. Web page. Last Accessed: 2017.02.25. URL: <https://neo4j.com/docs/operations-manual/current/clustering/> (cited on page 712).
- [442] Neo4j. *Chapter 4. Clustering*. Web page. Last Accessed: 2017.02.25. URL: <https://neo4j.com/docs/operations-manual/current/clustering/> (cited on page 712).

- [443] Neo4j. *Highly Available cluster*. Web page. Last Accessed: 2017.02.25. URL: <https://neo4j.com/docs/operations-manual/current/clustering/high-availability/architecture/> (cited on page 712).
- [444] *NetCDF: Introduction and Overview*. Web Page. URL: [https://www.unidata.ucar.edu/software/netcdf/docs\\_rc/](https://www.unidata.ucar.edu/software/netcdf/docs_rc/) (visited on 02/13/2017) (cited on page 717).
- [445] Netsyslab. *TOTEM*. Webpage. URL: <http://netsyslab.ece.ubc.ca/wiki/index.php/Totem> (cited on page 688).
- [446] *Netty Site*. Web Page. URL: <http://netty.io/> (cited on page 689).
- [447] *Nimbus*. Web Page. URL: <http://www.nimbusproject.org/doc/nimbus/platform/> (cited on page 744).
- [448] *Nimbus Wiki*. Web Page. URL: [https://en.wikipedia.org/wiki/Nimbus\\_\(cloud\\_computing\)](https://en.wikipedia.org/wiki/Nimbus_(cloud_computing)) (cited on page 743).
- [449] Ninefold. *Ninefold news*. Web Page. Accessed: 2017-02-27. Nov. 2015. URL: <http://ninefold.com/news/> (cited on page 669).
- [450] Nokia Corp. Web page. accessed 25-feb-2017. Feb. 2017. URL: [http://www.nokia.com/en\\_int](http://www.nokia.com/en_int) (cited on page 686).
- [451] Jordan Novet. *dotCloud*. Web Page. Accessed: 2017-1-31. Jan. 2016. URL: <http://venturebeat.com/2016/01/22/dotcloud-the-cloud-service-that-gave-birth-to-docker-is-shutting-down-on-february-29/> (cited on page 671).
- [452] *NSA 'NiFi' Big Data Automation Project Out In The Open*. Web Page. URL: <http://www.forbes.com/sites/adrianbridgwater/2015/07/21/nsa-nifi-big-data-automation-project-out-in-the-open/#6b37cac55d9a> (cited on page 651).
- [453] Daniel Nurmi et al. "The eucalyptus open-source cloud-computing system". In: *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society. 2009, pages 124–131 (cited on page 743).
- [454] Christian Nutt. *Gamasutra - Amazon launches new, free, high-quality game engine: Lumberyard*. URL: [http://www.gamasutra.com/view/news/265425/Amazon\\_launches\\_new\\_free\\_highquality\\_game\\_engine\\_Lumberyard.php](http://www.gamasutra.com/view/news/265425/Amazon_launches_new_free_highquality_game_engine_Lumberyard.php) (visited on 02/27/2017) (cited on page 679).
- [455] *NWB*. Web Page. 2017. URL: <http://nwb.cns.iu.edu/about.html> (cited on page 662).
- [456] *ODE Website*. Web Page. Accessed: 2017-02-11. URL: <http://ode.apache.org/> (cited on page 643).
- [457] *Apache ODE*. Web Page. Accessed: 2017-02-11. URL: [https://en.wikipedia.org/wiki/Apache\\_ODE](https://en.wikipedia.org/wiki/Apache_ODE) (cited on page 643).
- [458] John O'Hara. "Toward a commodity enterprise middleware". In: *Queue* 5.4 (2007), pages 48–55 (cited on page 690).
- [459] Carl W Olofson. *The Analytic-Transactional Data Platform: Enabling the Real-Time Enterprise (IDC)*. Technical report. Accessed: 2017-1-17. International Data Corporation (IDC), Dec. 2014. URL: <http://www.sap.com/documents/2016/08/3c4e546e-817c-0010-82c7-eda71af511fa.html> (cited on page 675).
- [460] Jean-Baptiste Onofre. *The Future of Apache Beam, Now a Top-Level Apache Software Foundation Project*. Blog. accessed 25-feb-2017. Jan. 2017. URL: <https://www.talend.com/blog/2017/01/13/future-apache-beam-now-top-level-apache-software-foundation-project/> (cited on page 753).
- [461] *Apache OODT*. Web Page. Accessed: 2017-02-25. URL: <http://oodt.apache.org/> (cited on page 754).
- [462] *Getting Started*. Web Page. Accessed: 2017-02-25. URL: <http://oodt.apache.org/documentation.htm> (cited on page 754).



- [463] Apache. *About OODT*. Web Page. Accessed: 2017-02-12. URL: <http://oodt.apache.org/> (cited on page 671).
- [464] *Oozie - Apache Oozie Workflow Scheduler for Hadoop*. Web Page. URL: <http://oozie.apache.org/> (visited on 02/13/2017) (cited on page 647).
- [465] *Open Database Connectivity*. Web Page. Accessed: 2017-1-30. URL: [https://en.wikipedia.org/wiki/Open\\_Database\\_Connectivity](https://en.wikipedia.org/wiki/Open_Database_Connectivity) (cited on page 699).
- [466] Open Source Robotics Foundation. *About ROS*. Web page. accessed 16-mar-2017. Mar. 2017. URL: <http://www.ros.org/about-ros/> (cited on page 757).
- [467] *OpenID*. website. URL: <http://openid.net/> (cited on page 749).
- [468] *OpenID*. webpage. URL: <https://en.wikipedia.org/wiki/OpenID> (cited on page 749).
- [469] *OpenNebula Wiki*. URL: <https://opennebula.org/about/technology/> (cited on page 743).
- [470] OpenRefine. *OpenRefine*. Web Page. accessed 2017-01-13. Feb. 2016. URL: <http://openrefine.org/> (cited on page 754).
- [471] OpenSFS, EOFS. *Welcome to the official home of the Lustre filesystem*. Web Page. accessed 2017-01-28. Dec. 2016. URL: <http://lustre.org/> (cited on page 727).
- [472] *OpenStack*. Web Page. Accessed:1/16/2017. URL: <https://www.openstack.org/> (cited on page 736).
- [473] *Keystone Architecture*. Web Page. Accessed: 2017-02-12. URL: <http://docs.openstack.org/developer/keystone/architecture.html> (cited on page 749).
- [474] *OpenVZ*. en. Web Page. Page Version ID: 764498783. Feb. 2017. URL: <https://en.wikipedia.org/w/index.php?title=OpenVZ&oldid=764498783> (visited on 02/13/2017) (cited on page 742).
- [475] Wikipedia. *Oracle Grid Engine - Wikipedia*. webpage. Accessed : 02-22-2017. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Oracle\\_Grid\\_Engine](https://en.wikipedia.org/wiki/Oracle_Grid_Engine) (cited on page 723).
- [476] *Oracle Database - Wikipedia*. Web Page. URL: [https://en.wikipedia.org/wiki/Oracle\\_Database](https://en.wikipedia.org/wiki/Oracle_Database) (visited on 02/13/2017) (cited on page 700).
- [477] *Oracle Labs PGX*. Web Page. Accessed: 2017-02-07. URL: [https://docs.oracle.com/cd/E56133\\_01/2.3.1/index.html](https://docs.oracle.com/cd/E56133_01/2.3.1/index.html) (cited on page 660).
- [478] *Oracle Website*. Web Page. Accessed: 2017-02-11. URL: <http://www.oracle.com/technetwork/database/database-technologies/berkeleydb> (cited on page 707).
- [479] Apache org. *Apache Tomcat*. Web Page. Accessed: 02-24-2017. URL: <http://tomcat.apache.org/> (cited on page 752).
- [480] G. Ostrouchov et al. Web Page. 2012. URL: <http://r-pbd.org/> (cited on page 654).
- [481] Todd Ouska. *Transport-level security tradeoffs using MQTT*. iot design website. Feb. 2016. URL: <http://iotdesign.embedded-computing.com/guest-blogs/transport-level-security-tradeoffs-using-mqtt/> (cited on pages 441, 442).
- [482] Overleaf. URL: <https://www.overleaf.com> (cited on page 149).
- [483] *Overview*. Online. The contact information listed on the webpage was for Yogesh Simmhan. URL: <http://dream-lab.cds.iisc.ac.in/about/overview/> (cited on page 661).
- [484] eclipse paho. *Python Client - documentation*. eclipse paho wensite. URL: <https://www.eclipse.org/paho/clients/python/docs/> (cited on pages 441, 444).
- [485] *Parallel Graph Analytics (PGX)*. Web Page. Accessed: 2017-02-11. URL: <http://www.oracle.com/technetwork/oracle-labs/parallel-graph-analytics/overview/index.html> (cited on page 660).
- [486] Parasol. Webpage. URL: <https://parasol.tamu.edu/research.php> (cited on page 661).

- [487] James J. (Jong Hyuk) Park et al. *Advanced Multimedia and Ubiquitous Engineering*. Edited by James J. (Jong Hyuk) Park et al. accessed 2017-02-13. Springer, 2016 (cited on page 686).
- [488] Mosha Pasumansky. *Inside Capacitor, BigQuery's next-generation columnar storage format*. Blog. Accessed: 2017-02-23. Apr. 2016. URL: <https://cloud.google.com/blog/big-data/2016/04/inside-capacitor-bigquerys-next-generation-columnar-storage-format> (cited on page 677).
- [489] PBS. Web Page. URL: <http://www.pbspro.org/> (cited on page 724).
- [490] Pegasus. Web Page. Accessed:2/3/2017. URL: <https://pegasus.isi.edu/> (cited on page 644).
- [491] *Pentaho*. webpage. URL: <https://en.wikipedia.org/wiki/Pentaho> (cited on page 652).
- [492] Francisco Perez-Sorrosal et al. *Omid's First Step in the Apache Community*. Web Page. accessed 2017-02-25. Sept. 2016. URL: <https://yahooeng.tumblr.com/post/151015726181/omids-first-step-in-the-apache-community> (cited on page 755).
- [493] Sebastian Peyrott. *API Gateway. An Introduction to Microservices, Part 2*. Website. Blog Post. Sept. 2015. URL: <https://auth0.com/blog/an-introduction-to-microservices-part-2-api-gateway/> (cited on page 756).
- [494] Rob Pike et al. "Interpreting the Data: Parallel Analysis with Sawzall". In: *Scientific Programming Journal* 13.4 (Oct. 2005), pages 277–298. ISSN: 1058-9244. DOI: 10.1155/2005/962135. URL: <https://research.google.com/archive/sawzall-sciprog.pdf> (cited on page 678).
- [495] Keshav Pingali et al. "The Tao of Parallelism in Algorithms". In: *The Tao of Parallelism in Algorithms*. Accessed: 2017-02-27. June 2011, pages 1–14. URL: <http://iss.ices.utexas.edu/Publications/Papers/pingali11.pdf> (cited on page 687).
- [496] Pivotal. Web Page. URL: <https://run.pivotal.io/features/> (cited on page 669).
- [497] *Pivotal Greenplum. The Open Source Massively Parallel Data Warehouse*. Webpage. URL: <https://pivotal.io/pivotal-greenplum> (cited on page 703).
- [498] *Pivotal Greenplum Database*. Webpage. URL: [https://en.wikipedia.org/wiki/Pivotal\\_Greenplum\\_Database](https://en.wikipedia.org/wiki/Pivotal_Greenplum_Database) (cited on page 703).
- [499] *Pivotal HAWQ*. URL: <http://hawq.incubator.apache.org/> (cited on page 676).
- [500] *Pivotal HDB*. URL: <https://pivotal.io/pivotal-hdb> (cited on page 676).
- [501] *PLASMA README*. Web Page. URL: <http://icl.cs.utk.edu/projectsfiles/plasma/html/README.html> (visited on 02/13/2017) (cited on page 655).
- [502] Moritz Plassnig. *Heroku-style Application Deployments with Docker - DZone Cloud*. Web Page. Accessed: 2017-1-17. Nov. 2015. URL: <https://dzone.com/articles/heroku-style-application-deployments-with-docker> (cited on page 738).
- [503] PmWiki. *GFFS - Global Federated File System*. Web Page. accessed 2017-01-28. Jan. 2012. URL: <http://genesis2.virginia.edu/wiki/Main/GFFS> (cited on page 728).
- [504] Ian Pointer. *Apache Beam want to be uber-API for big data*. Web page. accessed 25-feb-2017. Apr. 2016. URL: <http://www.infoworld.com/article/3056172/application-development/apache-beam-wants-to-be-uber-api-for-big-data.html> (cited on page 753).
- [505] *PolyBase*. Web Page. URL: [www.msdn.microsoft.com/en-us/library/mt143171.aspx](http://www.msdn.microsoft.com/en-us/library/mt143171.aspx) (cited on pages 675, 676).
- [506] Florin Pop, Joanna Kolodziej, and Beniamino DiMartino. "Resource Management for Big Data Platforms". In: Springer Nature, 2016, pages 49–50 (cited on page 649).
- [507] Florin Pop, Joanna Kolodziej, and Beniamino DiMartino. "Resource Management for Big Data Platforms". In: Springer Nature, 2016, pages 50–51 (cited on page 758).

- [508] *Potree*. URL: <http://potree.org/wp/about/> (cited on page 665).
- [509] Herbert Pötzl. *Linux-Vserver*. Web Page. URL: [www.linux-vserver.org](http://www.linux-vserver.org) (cited on page 742).
- [510] Vignesh Prajapati. *Big data analytics with R and Hadoop*. Packt Publishing, 2013. ISBN: 9781782163282 (cited on page 654).
- [511] Prashanth. *Facebook Tupperware*. Blog. Accessed: 2/18/2017. Dec. 2015. URL: <http://blog.cspp.in/index.php/2015/12/17/facebooks-tupperware/> (cited on page 736).
- [512] *Presto*. Web Page. Accessed: 2017-1-24. URL: <https://prestodb.io/> (cited on page 676).
- [513] *Project Jupyter*. Web page. URL: <http://www.jupyter.org> (visited on 02/27/2017) (cited on page 646).
- [514] UltraScan Project. *UltraScan Analysis Software*. Web page. Accessed: 2017-02-22. Apr. 2015. URL: <http://ultrascan.uthscsa.edu> (cited on page 644).
- [515] *Protocol Buffer*. Web Page. Sept. 2016. URL: <https://developers.google.com/protocol-buffers/> (cited on page 751).
- [516] Roshan Punnoose, Adina Crainiceanu, and David Rapp. “Rya: A Scalable RDF Triple Store for the Clouds”. In: *Proceedings of the 1st International Workshop on Cloud Intelligence*. Cloud-I ’12. Istanbul, Turkey: ACM, 2012, 4:1–4:8. ISBN: 978-1-4503-1596-8. DOI: 10.1145/2347673.2347677. URL: <http://doi.acm.org/10.1145/2347673.2347677> (cited on page 712).
- [517] *Puppet FAQ*. Web Page. URL: <https://puppet.com/product/faq> (cited on page 733).
- [518] *Puppet software*. Web Page. URL: [https://en.wikipedia.org/wiki/Puppet\\_\(software\)](https://en.wikipedia.org/wiki/Puppet_(software)) (cited on page 733).
- [519] PuppetLabsRazor. *Home.PuppetLabs/Razor Wiki*. Web Page. Accessed: 2017-02-04. URL: <https://github.com/puppetlabs/Razor/wiki> (cited on page 735).
- [520] PuppetLabsRazor. *Introducing Razor, a next Generation provisioning Solution-Puppet*. Web Page. Accessed: 2017-02-04. URL: <https://puppet.com/blog/introducing-razor-a-next-generation-provisioning-solution> (cited on page 735).
- [521] Qemu. *QEMU*. WebPage. Feb. 2017. URL: [http://wiki.qemu-project.org/index.php/Main\\_Page](http://wiki.qemu-project.org/index.php/Main_Page) (cited on page 741).
- [522] Quora. *LinkedIn*. Web page. accessed 18-March-2017. URL: <https://www.quora.com/What-is-LinkedIn-s-database-architecture-like> (cited on page 682).
- [523] *R: What is R?* Web Page. Accessed: 2017-1-26. URL: <https://www.r-project.org/about.html> (cited on page 654).
- [524] *RabbitMQ, components*. Web Page. Accessed: 2017-01-19. URL: <https://www.rabbitmq.com/> (cited on page 690).
- [525] Tilmann Rabl et al. “Solving Big Data Challenges for Enterprise Application Performance Management”. In: *Proc. VLDB Endow*. 5.12 (Aug. 2012), pages 1724–1735. ISSN: 2150-8097. DOI: 10.14778/2367502.2367512. URL: <https://arxiv.org/pdf/1208.4167.pdf> (cited on page 706).
- [526] Dan Radez. *OpenStack Essentials*. Packt Publishing Ltd., May 26, 2015. 182 pages. ISBN: 978-1-78398-708-5. URL: <http://ebook.konfigurasi.net/Openstack/OpenStack%5C%20Essentials.pdf> (cited on page 726).
- [527] Ioan Raicu et al. “Falkon: a Fast and Light-weight task executiON framework”. In: *ACM SC07*. 2007 (cited on page 725).
- [528] Siddharth Rao. *DREAM:Lab - Democratising Computing through the Cloud*. Online. Mar. 2016. URL: <http://iisc.researchmedia.center/article/dreamlab-%E2%80%93-democratising-computing-through-cloud> (cited on page 661).

- [529] *Rasdaman raster array database*. Web Page. Accessed: 02-23-2017. URL: <http://www.rasdaman.org/> (cited on page 703).
- [530] Raspberry Pi Foundation. *INSTALLING OPERATING SYSTEM IMAGES*. Web Page. accessed: 2018-1-20. URL: <https://www.raspberrypi.org/documentation/installation/installing-images/> (cited on pages 843, 844).
- [531] Raspberry Pi Foundation. *INSTALLING OPERATING SYSTEM IMAGES*. Web Page. accessed: 2018-1-20. URL: <https://www.raspberrypi.org/documentation/installation/installing-images/> (cited on pages 843, 844).
- [532] Raspberry Pi Foundation. *Lesson 3 - Dynamic Host Configuration Protocol (DHCP)*. Web Page. accessed: 2018-2-26. URL: <https://www.raspberrypi.org/learning/networking-lessons/lesson-3/plan/> (cited on pages 822, 827).
- [533] Raspberry Pi Foundation. *NOOBS*. Web Page. accessed: 2018-1-20. URL: <https://www.raspberrypi.org/documentation/installation/noobs.md> (cited on pages 843, 844).
- [534] Syed Ali Raza. *Neo4j*. Presentation/ Slides. Last Accessed: 2017.02.25. URL: <https://www.slideshare.net/aliraza995/neo4j-graph-storage-27104408> (cited on page 712).
- [535] *RCFile Cat*. Web Page. Accessed: 2017-02-02. URL: <https://cwiki.apache.org/confluence/display/Hive/RCFileCat> (cited on page 718).
- [536] *RCFileCat - Apache Hive*. Web Page. Accessed: 2017-1-27. URL: <https://cwiki.apache.org/confluence/display/Hive/RCFileCat> (cited on page 718).
- [537] Amazon RDS. *What is Amazon RDS*. Web Page. Accessed: 2017-02-04. URL: <http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html#Welcome.Concepts> (cited on page 704).
- [538] Amazon RDS. *What is Amazon RDS?Definition*. Web Page. Accessed: 2017-02-04. URL: <http://searchaws.techtarget.com/definition/Amazon-Relational-Database-Service-RDS> (cited on page 704).
- [539] Red Hat, Inc. *developers-openshift components*. Web Page. Accessed: 2017-1-26. URL: <https://developers.openshift.com/> (cited on page 667).
- [540] Red Hat, Inc. *openshift components*. Web Page. Accessed: 2017-1-26. URL: <https://www.openshift.org/> (cited on page 667).
- [541] Red Hat, Inc. *openshift-blog components*. Web Page. Accessed: 2017-1-27. URL: <https://blog.openshift.com/getting-started-with-openshift-origin-the-open-source-platform-as-a-service-paas/> (cited on page 667).
- [542] Red Hat, Inc. *paas-openshift components*. Web Page. Accessed: 2017-1-25. URL: <https://blog.openshift.com/announcing-openshift-origin-the-open-source-platform-as-a-service-paas/> (cited on page 667).
- [543] Red Hat, Inc. *Ansible Documentation*. Web Page. Accessed: 2017-02-12. Feb. 2015. URL: <https://docs.ansible.com/ansible/index.html> (cited on page 733).
- [544] Red Hat, Inc. *Ceph Homepage-Ceph*. Web Page. Accessed: 2017-1-26. Red Hat, Inc., 2017. URL: <https://ceph.com/> (cited on page 726).
- [545] Red Hat, Inc. *Ceph Filesystem - Ceph Documentation*. Web Page. Accessed: 2017-1-24. Red Hat, Inc. URL: <http://docs.ceph.com/docs/master/cephfs/> (cited on page 726).
- [546] Red Hat, Inc. *Ceph Architecture-Ceph Documentation*. Web Page. Accessed: 2017-1-24. 2017. URL: <http://docs.ceph.com/docs/master/architecture/> (cited on page 726).
- [547] Bloor Robin and Jozwiak Rebecca. *THE NATURE OF GRAPH DATA*. Technical report. Austin, TX 78720|512-524-3689: The Bloor Group, 2017. URL: <http://web.cray.com/bloor-graph-data> (cited on page 713).



- [548] *Rocks*. Web Page. 2017. URL: <https://www.rockscluster.org> (cited on page 736).
- [549] Ryan Rosario. *Exciting Tools for Big Data: S4, Sawzall and mrjob!* Web page. Online; accessed 30-jan-2017. Nov. 2010. URL: <http://www.bytemining.com/2010/11/exciting-tools-for-big-data-s4-sawzall-and-mrjob/> (cited on page 678).
- [550] Margaret Rouse and Jose C Elias. *ACID (atomicity, consistency, isolation, and durability)*. Website. July 2006. URL: <http://searchsqlserver.techtarget.com/definition/ACID> (cited on page 701).
- [551] Apache Rya. *Apache Rya*. Online. URL: <https://rya.apache.org/> (cited on page 712).
- [552] *S4: S4 0.6.0 overview*. Web Page. Accessed: 2017-02-24. URL: <http://incubator.apache.org/s4/doc/0.6.0/overview> (cited on page 680).
- [553] *Sahara*. Web Page. Accessed: 1/16/2017. URL: <http://docs.openstack.org/developer/sahara/> (cited on page 736).
- [554] *saltstack components*. webpage. accessed: 2017-02-4. URL: <https://saltstack.com/> (cited on page 734).
- [555] *Samza*. Web Page. URL: <http://samza.apache.org/> (visited on 02/13/2017) (cited on page 680).
- [556] Abhijeet Sandil. *SSO Strategy: Authentication (SAML) -vs- Authorization (OAuth)*. Web Page. Accessed: 2017-02-04. URL: <https://www.linkedin.com/pulse/sso-strategy-authentication-vs-authorization-saml-oauth-sandil> (cited on page 749).
- [557] Lee Sangkeun, Rangan Sukumar Sreenivas, and Lim Seung-Hwan. “Graph mining meets the semantic web.” In: *2015 31st IEEE International Conference on Data Engineering Workshops*. (Seoul, South Korea). Edited by Johannes Gehrke, Wolfgang Lehner, and Kyuseok Shim. IEEE. Apr. 2015, pages 53–58. DOI: 10.1109/ICDEW.2015.7129544. URL: [https://www.researchgate.net/profile/Sreenivas\\_Rangan\\_Sukumar/publication/273755615\\_Graph\\_Mining\\_Meets\\_the\\_Semantic\\_Web/links/550a27cb0cf20ed529e26260.pdf](https://www.researchgate.net/profile/Sreenivas_Rangan_Sukumar/publication/273755615_Graph_Mining_Meets_the_Semantic_Web/links/550a27cb0cf20ed529e26260.pdf) (cited on page 713).
- [558] Shilpi Saxena and Sumit Gupta. *Real-Time Big Data Analytics*. 1st. 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing, 2016. ISBN: 9781784391409 (cited on page 681).
- [559] netlib. *ScaLAPACK Users Guide*. Web Page. Accessed: 2017-02-12. URL: <http://www.netlib.org/scalapack/slug/> (cited on page 655).
- [560] *Scaling Apache Giraph to a trillion edges*. web page. URL: <https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to-a-trillion-edges/10151617006153920> (cited on page 686).
- [561] Tom Schaul et al. “PyBrain”. In: *Journal of Machine Learning Research* 11 (Mar. 2010). Edited by Soeren Sonnenburg, pages 743–746. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=1756006.1756030> (cited on page 657).
- [562] SchedMd. *Slurm website*. Web Page. Accessed: 2017-02-27. Mar. 2013. URL: <https://slurm.schedmd.com/overview.html> (cited on page 724).
- [563] SchedMd. *Slurm Supported Platforms*. Web Page. Accessed: 2017-02-27. Apr. 2015. URL: <https://slurm.schedmd.com/platforms.html> (cited on page 724).
- [564] *scikit-Learn*. webpage. URL: <https://en.wikipedia.org/wiki/Scikit-learn> (cited on page 657).
- [565] Sean Lynch. *Why Membase Uses Erlang*. web page. accessed 2017-01-29. Oct. 2010. URL: <https://blog.couchbase.com/why-membase-uses-erlang%7D> (cited on page 709).
- [566] Adam Seligman. *Apache Phoenix - A Small Step for Big Data*. Web page. Online; accessed 25-jan-2017. May 2014. URL: <https://developer.salesforce.com/blogs/>

- [developer-relations/2014/05/apache-phoenix-small-step-big-data.html](#) (cited on page 674).
- [567] James Serra. *What is Azure Data Factory?* Web Page. Accessed: 02/03/2016. URL: <http://www.jamesserra.com/archive/2014/11/what-is-azure-data-factory/> (cited on page 650).
- [568] Vinay Jayarama Setty. “Publish/subscribe for large-scale social interaction: Design, analysis and resource provisioning”. Accessed:2017-1-29. PhD thesis. Faculty of Mathematics and Natural Sciences, University of Oslo: Department of Informatics, UNIVERSITY OF OSLO, Jan. 2015. URL: <https://www.duo.uio.no/bitstream/handle/10852/43117/1595-Setty-DUO-Thesis.pdf?sequence=1&isAllowed=y> (cited on page 688).
- [569] ShareLaTeX. URL: <https://www.sharelatex.com> (cited on page 149).
- [570] Caro Caserio Sharon Lo Sreedhar Pelluru. *Introduction to Azure Data Factory Service, a data integration service in the cloud*. Web Page. Accessed: 02/03/2016. URL: <https://docs.microsoft.com/en-us/azure/data-factory/data-factory-introduction> (cited on page 650).
- [571] Xuanhua Shi et al. “GIRAFFE: A Scalable Distributed Coordination Service for Large-scale Systems”. In: *GIRAFFE: A Scalable Distributed Coordination Service for Large-scale Systems*. Accessed: 2017-02-27. Sept. 2014, pages 1–10. URL: <http://www.mcs.anl.gov/papers/P5157-0714.pdf> (cited on page 750).
- [572] Abdul Ghaffar Shoro and Tariq Rahim Soomro. “Big data analysis: Apache spark perspective”. In: *Global Journal of Computer Science and Technology* 15.1 (2015). Accessed: 2017-1-21. ISSN: 0975-4172. URL: <http://www.computerresearch.org/index.php/computer/article/view/1137/1124> (cited on pages 653, 685).
- [573] Pallickara Shrideep et al. *granules*. Project. July 2016. URL: <http://granules.cs.colostate.edu/> (cited on page 681).
- [574] Julian Shun and Guy E. Blelloch. “Ligra: A Lightweight Graph Processing Framework for Shared Memory”. In: *ACM SIGPLAN Notices - PPOPP '13*. Pittsburgh, Pennsylvania, USA: ACM New York, NY, USA, 2013. ISBN: 978-1-4503-1922-5. DOI: 10.1145/2517327.2442530. URL: <http://dl.acm.org/citation.cfm?id=2442530> (cited on page 687).
- [575] Jeff Shute et al. “F1: The Fault-tolerant Distributed RDBMS Supporting Google’s Ad Business”. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. SIGMOD '12. Accessed : 2017-02-15. Scottsdale, Arizona, USA: ACM, 2012, pages 777–778. ISBN: 978-1-4503-1247-9. DOI: 10.1145/2213836.2213954. URL: <http://doi.acm.org/10.1145/2213836.2213954> (cited on page 704).
- [576] Jeff Shute et al. “F1: A Distributed SQL Database That Scales”. In: *Proc. VLDB Endow.* 6.11 (Aug. 2013). Accessed:2017-02-15, pages 1068–1079. ISSN: 2150-8097. DOI: 10.14778/2536222.2536232. URL: <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41344.pdf> (cited on page 704).
- [577] Abraham Silberschatz et al. *Operating system concepts*. Volume 4. Addison-wesley Reading, 1998 (cited on page 694).
- [578] SINTEF. *CloudML*. Web page. accessed 15-March-2017. 2013. URL: <http://cloudml.org/> (cited on page 739).
- [579] SINTEF Git. *CloudML Wiki*. Web page. accessed 16-March-2017. Last updated in 2 Oct 2015. URL: <https://github.com/SINTEF-9012/cloudml/wiki> (cited on page 739).
- [580] Mark Slee, Aditya Agarwal, and Marc Kwiatkowski. “Thrift: Scalable Cross-Language Services Implementation”. In: *Thrift* (Apr. 1, 2007). URL: <https://thrift.apache.org/static/files/thrift-20070401.pdf> (cited on page 751).

- [581] Slideshare. *Apache UIMA Introduction*. Web Page. Accessed: 02/03/2016. URL: <http://www.slideshare.net/teofili/apache-uima-introduction> (cited on page 699).
- [582] SLURM. Web Page. URL: <https://slurm.schedmd.com/> (cited on page 724).
- [583] Ken Smith. *Azure: What to Use, What to Avoid*. Web page. Online; accessed 28-jan-2017. Oct. 2014. URL: <http://blog.wouldbetheologian.com/2014/10/azure-what-to-use-what-to-avoid.html> (cited on page 715).
- [584] Russell Smith. *What are the advantages and disadvantages of using MongoDB vs CouchDB vs Cassandra vs Redis?* Web page. Online; accessed 29-jan-2017. Nov. 2015. URL: <https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-MongoDB-vs-CouchDB-vs-Cassandra-vs-Redis> (cited on page 709).
- [585] Amazon SNS. Web Page. Accessed: 2017-02-02. URL: <https://aws.amazon.com/sns/> (cited on page 693).
- [586] *Amazon SNS Blog*. Web Page. Accessed: 2017-02-02. URL: <https://aws.amazon.com/blogs/aws/introducing-the-amazon-simple-notification-service/> (cited on page 693).
- [587] *Amazon SNS FAQs*. Web Page. Accessed: 2017-02-02. URL: <https://aws.amazon.com/sns/faqs/> (cited on page 693).
- [588] *Software >> OpenStack Open Source Cloud Computing Software*. Web Page. Online; accessed 23-Feb-2017. URL: <https://www.openstack.org/software/> (cited on page 743).
- [589] Borja Sotomayor and Lisa Childers. *Globus® Toolkit 4: Programming Java Services*. Morgan Kaufmann, 2006 (cited on page 724).
- [590] Apache Software Foundation. *Spark Programming Guide - Spark 2.1.0 Documentation*. Web Page. Accessed: 04-09-2017. URL: <http://spark.apache.org/docs/latest/programming-guide.html#resilient-distributed-datasets-rdds> (cited on page 683).
- [591] Evan R Sparks et al. “KeystoneML: Optimizing Pipelines for Large-Scale Advanced Analytics”. In: *arXiv preprint arXiv:1610.09451* (2016) (cited on page 652).
- [592] Evan Sparks and Shivaram Venkataraman. *Building Large Scale Machine Learning Applications with Pipelines*. URL: <https://www.youtube.com/watch?v=30r8sna79ms> (visited on 02/27/2017) (cited on page 652).
- [593] E. Sparks et al. “MLI: An API for Distributed Machine Learning”. In: *MLI: An API for Distributed Machine Learning*. IEEE 13th International Conference on Data Mining. 2013 (cited on page 653).
- [594] Apache Software Foundation. *Spark Streaming - Spark 2.1.0 Documentation*. Web Page. Accessed: 04-09-2017. URL: <http://spark.apache.org/docs/latest/streaming-programming-guide.html> (cited on page 683).
- [595] Kenneth Spencer. *Configuring the Raspberry Pi as a DHCP Server under Raspbian Wheezy*. Web Page. accessed: 2018-2-26. Mar. 2013. URL: [http://www.my-music.mine.nu/images/rpi\\_raspbianwheezy\\_dhcp\\_server.pdf](http://www.my-music.mine.nu/images/rpi_raspbianwheezy_dhcp_server.pdf) (cited on pages 822, 823, 826, 827).
- [596] Splunk, Inc. *Splunk Enterprise Overview*. Web Page. accessed: 2017-02-18. Feb. 2015. URL: <http://docs.splunk.com/Documentation/Splunk/6.5.2/Overview/AboutSplunkEnterprise> (cited on page 664).
- [597] SQLite. *About SQLite*. Website. URL: <https://www.sqlite.org/about.html> (cited on page 701).
- [598] SQLite. *Appropriate Uses For SQLite*. Website. URL: <https://www.sqlite.org/whentouse.html> (cited on page 701).
- [599] *SQL Server Wiki*. Web Page. URL: [https://en.wikipedia.org/wiki/Microsoft\\_SQL\\_Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) (cited on page 700).



- [600] *SSH - Wikipedia*. Web Page. Accessed: 2017-01-26. URL: [https://en.wikipedia.org/wiki/Secure\\_Shell](https://en.wikipedia.org/wiki/Secure_Shell) (cited on page 719).
- [601] *OpenSSH - Wikipedia*. Web Page. Accessed: 2017-01-26. URL: <https://en.wikipedia.org/wiki/OpenSSH> (cited on page 719).
- [602] *Difference between Application Manager and Application Master in YARN?* StackOverflow. 2015. URL: <http://stackoverflow.com/questions/30967247/difference-between-application-manager-and-application-master-in-yarn> (cited on page 721).
- [603] Michael Stonebraker. “SciDB: An Open-Source DBMS for Scientific Data”. In: *ERCIM News* 89 (Apr. 2012), page 56. URL: <https://ercim-news.ercim.eu/en89/special/scidb-an-open-source-dbms-for-scientific-data> (cited on page 702).
- [604] *Apache Storm - Concepts*. webpage. Accessed : 01-22-2017. URL: <http://storm.apache.org/releases/1.0.2/Concepts.html> (cited on page 680).
- [605] Apache storm. *Storm MQTT Integration*. Apache storm website. URL: <http://storm.apache.org/releases/1.1.0/storm-mqtt.html> (cited on page 442).
- [606] Seed Studio. *Grove Relay*. seed studio website. Oct. 2017. URL: <http://wiki.seeed.cc/Grove-Relay/> (cited on page 444).
- [607] seed studio. *Grove LED socket kit*. Seed studio website. Oct. 2017. URL: [http://wiki.seeed.cc/Grove-LED\\_Socket\\_Kit/](http://wiki.seeed.cc/Grove-LED_Socket_Kit/) (cited on page 444).
- [608] Dan Sullivan. *Paas Provider Comparison Guide: Engine Yard - Orchestration and Management*. Article. Accessed: 02-26-2017. July 2013. URL: <http://www.tomsitpro.com/articles/paas-engine-yard-amazon-elastic-cloud-computing,2-578.html> (cited on page 670).
- [609] Nick Sullivan. *Kyoto Tycoon Secure Replication*. Web Page. Accessed: 02/03/2016. URL: <https://blog.cloudflare.com/kyoto-tycoon-secure-replication/> (cited on page 707).
- [610] Bruce Synder and Rob Davies. *ActiveMQ in Action*. 1st. Manning publications, 2013. ISBN: 1933988940, 9781933988948 (cited on page 690).
- [611] Systap. *Blazegraph Wiki*. Web Page. Page was last modified on 25 Jan 2016. Aug. 2008. URL: [https://wiki.blazegraph.com/wiki/index.php/Main\\_Page](https://wiki.blazegraph.com/wiki/index.php/Main_Page) (cited on page 714).
- [612] Istvan Szegedi. *Apache Phoenix - An SQL Driver for HBase*. Web page. Online; accessed 23-jan-2017. May 2014. URL: <https://bighadoop.wordpress.com/2014/05/17/apache-phoenix-an-sql-driver-for-hbase/> (cited on page 674).
- [613] *Tableau Tutorial*. Web Page. Accessed: 2017-02-10. URL: <https://casci.umd.edu/wp-content/uploads/2013/12/Tableau-Tutorial.pdf> (cited on page 664).
- [614] *Tableau Technology*. Web Page. Accessed: 2017-02-10. URL: <https://www.tableau.com/products/technology> (cited on page 664).
- [615] Niranjana Tallapalli. *Hadoop and Spark Installation on Raspberry Pi3 Cluster Part 3*. Web Page. accessed: 2018-2-26. Feb. 2017. URL: <https://tekmarathon.com/2017/02/16/hadoop-and-spark-installation-on-raspberry-pi-3-cluster-part-3/> (cited on page 827).
- [616] A. Talwalkar et al. “MLbase: A Distributed Machine Learning Wrapper”. In: *MLbase: A Distributed Machine Learning Wrapper*. Big Learning Workshop at NIPS. 2012 (cited on page 653).
- [617] *Taverna*. Web Page. URL: <https://taverna.incubator.apache.org/introduction/> (cited on page 645).

- [618] Illinois Institute of Technology Department of Computer Science. *ZHT: a zero-hop distributed hashtable*. Online. URL: <http://datasys.cs.iit.edu/projects/ZHT/> (cited on page 706).
- [619] Techopedia. *Amazon Web Services (AWS)*. Web Page. URL: <https://www.techopedia.com/definition/26426/amazon-web-services-aws> (cited on page 745).
- [620] TechTarget. *Lightweight Directory Access Protocol*. Web Page. accessed 2017-01-30. Feb. 2017. URL: <http://searchmobilecomputing.techtarget.com/definition/LDAP> (cited on page 749).
- [621] *TensorFlow*. Web Page. Accessed: 2017-1-24. URL: <https://www.tensorflow.org> (cited on page 666).
- [622] *Terraform components*. Web Page. Accessed: 2017-02-12. URL: <https://www.terraform.io/intro/index.html> (cited on page 739).
- [623] *Tez*. Web Page. 2017. URL: <https://hortonworks.com/apache/tez> (cited on page 648).
- [624] The Apache Software Foundation. Web Page. URL: <http://sqoop.apache.org/> (cited on page 720).
- [625] The Apache Software Foundation. Web Page. URL: <http://ant.apache.org/> (cited on page 755).
- [626] The Apache Software Foundation. *Apache Flume*. web page. accessed 2017-02-06. URL: <https://flume.apache.org/index.html> (cited on page 720).
- [627] The Apache Software Foundation. *Getting Started with Derby*. Web Page. accessed: 2017-02-18. Sept. 2015. URL: <https://db.apache.org/derby/docs/10.12/getstart/index.html> (cited on page 703).
- [628] The Apache Software Foundation. *Apache Derby*. Web Page. accessed: 2017-02-18. Oct. 2016. URL: <https://db.apache.org/derby/> (cited on page 703).
- [629] The Apache Software Foundation. *Apache Derby Project Charter*. Web page. accessed: 2017-02-18. Sept. 2016. URL: [https://db.apache.org/derby/derby\\_charter.html](https://db.apache.org/derby/derby_charter.html) (cited on page 703).
- [630] The Apache Software Foundation. *Spark SQL*. Web Page. accessed 2017-02-19. Feb. 2016. URL: <http://spark.apache.org/sql/> (cited on page 705).
- [631] The Apache Software Foundation. *Apache CloudStack*. Web Page. Accessed: 2017-02-11. Feb. 2017. URL: <https://cloudstack.apache.org/> (cited on page 744).
- [632] The Disco Project. *What is Disco*. Web page. accessed 25-feb-2017. Feb. 2017. URL: <http://disco.readthedocs.io/en/develop/intro.html> (cited on page 686).
- [633] The Disco Project. *Why Not Hadoop?* Web page. accessed 25-feb-2017. Feb. 2017. URL: <http://disco.readthedocs.io/en/develop/faq.html#why-not-hadoop> (cited on page 686).
- [634] *The Eucalyptus Open-Source Private Cloud*. Web Page. Accessed: 2017-02-11. URL: <http://www.cloudbook.net/resources/stories/the-eucalyptus-open-source-private-cloud> (cited on page 743).
- [635] *The Jupyter notebook --- Jupyter Notebook 5.0.0.dev documentation*. Web page. URL: <https://jupyter-notebook.readthedocs.io/en/latest/index.html> (visited on 02/27/2017) (cited on page 646).
- [636] *The LXD container hypervisor*. web page. URL: <https://www.ubuntu.com/containers/lxd> (cited on page 755).
- [637] The Open MPI Project. *Open MPI: Open Source High Performance Computing*. Web Page. Accessed: 2017-02-22. Feb. 2017. URL: <https://www.open-mpi.org> (cited on page 752).

- [638] The PostgreSQL Global Development Group. *PostgreSQL 9.5: UPSERT, Row Level Security, and Big Data*. Web Page. Accessed: 2017-4-10. URL: <https://www.postgresql.org/about/> (cited on page 701).
- [639] The PostgreSQL Global Development Group. *PostgreSQL 9.5: UPSERT, Row Level Security, and Big Data*. Web Page. Accessed: 2017-4-10. URL: <https://www.postgresql.org/about/news/1636/> (cited on page 702).
- [640] The Windows Club. *Understanding Blob, Queue and Table Storage for Windows Azure*. Web page. Online; accessed 28-jan-2017. Jan. 2017. URL: <http://www.thewindowsclub.com/understanding-blobqueue-table-storage-windows-azure> (cited on page 715).
- [641] *theano*. Web Page. Accessed: 2017-1-21. URL: <http://deeplearning.net/software/theano/introduction.html> (cited on page 659).
- [642] Chandu Thekkath. *Naiad - Microsoft Research*. webpage. Accessed: 2017-1-27. Microsoft, Oct. 2011. URL: <https://www.microsoft.com/en-us/research/project/naiad/> (cited on page 647).
- [643] *Three.js*. Web Page. Accessed: 2017-02-27. Mar. 2017. URL: <https://en.wikipedia.org/wiki/Three.js> (cited on page 665).
- [644] *Creating a scene*. Web Page. Accessed: 2017-02-27. Dec. 2016. URL: [https://threejs.org/docs/index.html#Manual/Getting\\_Started/Creating\\_a\\_scene](https://threejs.org/docs/index.html#Manual/Getting_Started/Creating_a_scene) (cited on page 665).
- [645] Apache. *About Thrift*. Web Page. Accessed: 2017-02-12. URL: <https://thrift.apache.org/> (cited on page 751).
- [646] *Apache Tika*. Web Page. URL: <https://tika.apache.org/> (cited on page 700).
- [647] *TinkerPop*. Web Page. Accessed: 2/6/2017. URL: <http://tinkerpop.apache.org/> (cited on page 714).
- [648] *Titan DB*. Web Page. Accessed: 2/4/2017. URL: <http://titan.thinkaurelius> (cited on page 714).
- [649] *Tokyo Cabinet*. Web Page. Accessed: 2017-1-27. URL: <http://fallabs.com/tokyocabinet/> (cited on page 707).
- [650] Stanimire Tomov, Jack Dongarra, and Marc Baboulin. “Towards dense linear algebra for hybrid GPU accelerated manycore systems”. In: *Parallel Computing* 36.5 (2010), pages 232–240. URL: <http://www.sciencedirect.com/science/article/pii/S0167819109001276> (visited on 02/13/2017) (cited on page 655).
- [651] Stanimire Tomov et al. “Dense linear algebra solvers for multicore with GPU accelerators”. In: *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. IEEE, 2010, pages 1–8. URL: [http://link.springer.com/chapter/10.1007/978-3-319-06548-9\\_1](http://link.springer.com/chapter/10.1007/978-3-319-06548-9_1) (visited on 02/13/2017) (cited on page 655).
- [652] Cardiff University. *Triana Documentation*. Web Page. Accessed: 2017-02-20. Cardiff University, 2012. URL: <http://www.trianacode.org/> (cited on page 645).
- [653] Craig Trim. *Jen: Semantic Web Framework*. Web Page. Accessed: 02/03/2016. URL: <http://trimc-nlp.blogspot.com/2013/06/introduction-to-jena.html> (cited on pages 714, 715).
- [654] *Taverna Workflows: Syntax and Semantics*. IEEE, Dec. 2007. ISBN: 0-7695-3064-8. DOI: 10.1109/E-SCIENCE.2007.71. URL: <http://ieeexplore.ieee.org/document/4426917/> (cited on page 645).
- [655] Matteo Turilli, Mark Santcroos, and Shantenu Jha. “A Comprehensive Perspective on Pilot-Job Systems”. In: *ACM arXiv*. Mar. 2016, pages 1–26 (cited on page 725).
- [656] James Turnbull. *The Terraform Book*. 110th edition. 9780988820258. Turnbull Press, Nov. 2016 (cited on page 739).

- [657] random nerds tutorial. *What is MQTT and How It Works*. random nerds website. URL: <https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/> (cited on page 440).
- [658] tutorialspoint. *SQLite Overview*. Website. URL: [https://www.tutorialspoint.com/sqlite/sqlite\\_overview.htm](https://www.tutorialspoint.com/sqlite/sqlite_overview.htm) (cited on page 701).
- [659] Microsoft Azure - *Queues*. Web Page. Accessed: 2017-02-10. URL: [https://www.tutorialspoint.com/microsoft\\_azure/microsoft\\_azure\\_queues.htm](https://www.tutorialspoint.com/microsoft_azure/microsoft_azure_queues.htm) (cited on page 694).
- [660] Dylan Tweney. *AppFog gives developers an easier way to deploy cloud apps (interview)*. Online. May 2012. URL: <http://venturebeat.com/2012/05/15/appfog-gives-developers-an-easier-way-to-deploy-cloud-apps-interview/> (cited on page 670).
- [661] *Twister: Iterative MapReduce*. Web Page. URL: <http://www.iterativemapreduce.org/> (visited on 02/13/2017) (cited on page 685).
- [662] Twitter. Web Page. Accessed: 2017-02-04. URL: <https://blog.twitter.com/2016/open-sourcing-twitter-heron> (cited on page 682).
- [663] Twitter. *Flying faster with Twitter Heron*. Web Page. Accessed: 2017-02-04. URL: <https://blog.twitter.com/2015/flying-faster-with-twitter-heron> (cited on page 682).
- [664] *Kyoto Tycoon*. Web Page. URL: <http://fallabs.com/kyototycoon/> (cited on page 708).
- [665] *Tyrant FALLabs*. Web Page. URL: <http://fallabs.com/tokyotyrant/> (cited on page 708).
- [666] *Tyrant Blog*. Web Page. URL: <https://www.percona.com/blog/2009/10/19/mysql-memcached-tyrant-part3/> (cited on page 708).
- [667] *Understanding LXC and LXD*. web page. URL: <http://thevarguy.com/open-source-application-software-companies/understanding-lxc-and-lxd-canonicals-open-source-containe> (cited on page 755).
- [668] ISS team - University of Texas. *Galois website*. Web Page. Accessed: 2017-02-27. URL: <http://iss.ices.utexas.edu/?p=projects/galois> (cited on page 687).
- [669] *University Policies: Cheating and Plagiarism, ACA-72*. Web Page. updates 1975. 1961. URL: <https://policies.iu.edu/policies/aca-72-cheating-plagiarism/index.html> (cited on pages 146, 147).
- [670] *Using Amazon S3*. Web Page. Accessed: 2017-1-27. URL: <http://docs.aws.amazon.com/AmazonS3/latest/gsg/CopyingAnObject.html> (cited on page 728).
- [671] Devananda Van Der Veen and Llama Wrangler. *Introduction to Ironic*. Web Page. Accessed: 2017-02-27. Mar. 2015. URL: <https://docs.openstack.org/developer/ironic/deploy/user-guide.html> (cited on page 737).
- [672] Venkat Venkataramani. *What is the TAO cache used for at Facebook*. Web Page. June 2013. URL: <https://www.quora.com/What-is-the-TAO-cache-used-for-at-Facebook> (cited on page 714).
- [673] Magister Vis. *What Is Lumberyard? (Lumberyard Tutorials Series #1) - YouTube*. URL: <https://www.youtube.com/watch?v=Fxwo3KqSsUI> (visited on 02/27/2017) (cited on page 679).
- [674] VMware. web-page. accessed 2017-02-13. URL: <http://www.vmware.com/products/esxi-and-esx.html> (cited on page 745).
- [675] vmware. *vCloud*. Webpage. URL: <http://www.vmware.com/products/vcloud-suite.html> (cited on page 745).

- [676] LinkedIn. *Project Voldemort*. Web Page. Accessed: 2017-1-17. URL: <http://www.project-voldemort.com/voldemort/> (cited on page 706).
- [677] VoltDB. Web Page. URL: <https://www.wired.com/2016/04/kites-coding-assitant-spots-errors-finds-better-open-source/> (cited on page 672).
- [678] VoltDB. Web Page. URL: <https://www.voltdb.com/> (cited on page 697).
- [679] w3. *Apache Jena*. Web Page. Accessed: 2016.02.03. URL: [https://www.w3.org/2001/sw/wiki/Apache\\_Jena](https://www.w3.org/2001/sw/wiki/Apache_Jena) (cited on page 714).
- [680] Liya Wang, Peter Van Buren, and Doreen Ware. “Architecting a distributed bioinformatics platform with iRODS and iPlant Agave API”. In: *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. Dec. 2015. ISBN: 9781467397957 (cited on pages 671, 672).
- [681] Lizhe Wang, Wei Jie, and Jinjun Chen. *Grid Computing: Infrastructure, Service, and Applications*. 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487: Taylor & Francis, 2009. ISBN: 1420067664 (cited on page 747).
- [682] Pau Kiat Wee. “Instant AppFog”. In: Packt Publishing, July 2013. Chapter 1. URL: <https://www.packtpub.com/mapt/book/Web-Development/9781782167624/1/ch01lv11sec03/So,+what+is+AppFog%3F> (cited on page 670).
- [683] Sage A. Weil et al. “Ceph: A scalable, high-performance distributed file system”. In: *Proceedings of the 7th symposium on Operating systems design and implementation*. OSDI ’06. Accessed: 2017-1-26. Seattle, Washington: USENIX Association, Nov. 2006, pages 307–320. ISBN: 1-931971-47-1. URL: [https://www.usenix.org/legacy/event/osdi06/tech/full\\_papers/weil/weil.pdf](https://www.usenix.org/legacy/event/osdi06/tech/full_papers/weil/weil.pdf) (cited on page 726).
- [684] Johannes Wettinger. *any2api - The better way to create awesome APIs*. Web Page. Accessed: 2017-02-02. URL: <http://www.any2api.org/> (cited on page 740).
- [685] Johannes Wettinger, Uwe Breitenbücher, and Frank Leymann. “DevOpSlang - Bridging the Gap Between Development and Operations”. In: *Proceedings of the 3rd European Conference on Service-Oriented and Cloud Computing (ESOCC 2014)*. Lecture Notes in Computer Science (LNCS). accessed 2017-02-26. Springer-Verlag, 2014, pages 108–122 (cited on page 740).
- [686] Johannes Wettinger, Uwe Breitenbücher, and Frank Leymann. “Any2API - Automated APIfication”. In: *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science*. Edited by Markus Helfert, Donald Ferguson, and Víctor Méndez Muñoz. Lisbon, Portugal: SciTePress, 20-22 May 2015, pages 475–486. DOI: 10.5220/0005472704750486. URL: <https://pdfs.semanticscholar.org/1cd4/4b87be8cf68ea5c4c642d38678a7b40a86de.pdf> (cited on page 740).
- [687] *What is a public cloud?* Webpage. URL: <http://www.interoute.com/cloud-article/what-public-cloud> (cited on page 746).
- [688] *What Is Amazon Route 53?* Web Page. Accessed: 2017-02-24. URL: <http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html> (cited on page 746).
- [689] *What is HTCondor?* Web Page. URL: <https://research.cs.wisc.edu/htcondor/description.html> (cited on page 723).
- [690] *What is hubzero?* Web Page. URL: <https://hubzero.org/documentation/1.0.0/installation> (cited on page 671).
- [691] *What is the Jupyter Notebook? --- Jupyter Notebook 5.0.0.dev documentation*. Web page. (Visited on 02/27/2017) (cited on page 646).
- [692] *Whirr technology*. webpage. accessed: 2017 - 2 - 4. URL: <https://whirr.apache.org/> (cited on page 732).
- [693] *Whirr technology*. webpage. accessed: 2017 - 2 - 4. URL: <http://www.slideshare.net/huguk/apache-whirr> (cited on page 732).



- [694] Why Oozie? | Just a simple Hadoop DBA. Web Page. URL: <https://prodlife.wordpress.com/2013/12/09/why-oozie/> (visited on 02/13/2017) (cited on page 647).
- [695] Why use LXC (Linux Containers) ? web page. URL: <http://www.jpablo128.com/why-use-lxc-linux-containers/> (cited on page 742).
- [696] Wikipedia. *Apache Flex*. Web Page. Accessed: 2017-03-06. Mar. 2017. URL: [https://en.wikipedia.org/wiki/Apache\\_Flex](https://en.wikipedia.org/wiki/Apache_Flex) (cited on page 757).
- [697] Wikipedia. Web Page. URL: <https://en.wikipedia.org/wiki/Sqoop> (cited on page 720).
- [698] Wikipedia. Web Page. URL: [https://en.wikipedia.org/wiki/Portable\\_Batch\\_System](https://en.wikipedia.org/wiki/Portable_Batch_System) (cited on page 723).
- [699] Wikipedia. web-page. accessed 2017-02-13. URL: [https://en.wikipedia.org/wiki/VMware\\_ESXi](https://en.wikipedia.org/wiki/VMware_ESXi) (cited on page 745).
- [700] Wikipedia. *Apache Apex wiki*. Web Page. Accessed: 2017-02-26. URL: [https://en.wikipedia.org/wiki/Apache\\_Apex](https://en.wikipedia.org/wiki/Apache_Apex) (cited on page 756).
- [701] Wikipedia. *DataNucleus Wiki*. Web Page. Accessed: 2017-02-04. URL: <https://en.wikipedia.org/wiki/DataNucleus> (cited on page 699).
- [702] Wikipedia. *Java Message Service - Wikipedia*. webpage. Accessed : 01-18-2017. URL: [https://en.wikipedia.org/wiki/Java\\_Message\\_Service](https://en.wikipedia.org/wiki/Java_Message_Service) (cited on page 692).
- [703] Wikipedia. *Kubernetes Wiki*. Web Page. URL: <https://en.wikipedia.org/wiki/Kubernetes> (cited on page 737).
- [704] Wikipedia. *Neo4j*. Web page. Last Accessed: 2017.02.25. URL: <https://en.wikipedia.org/wiki/Neo4j> (cited on page 712).
- [705] Wikipedia. *UIMA*. Web Page. Accessed: 2017.02.03. URL: <https://en.wikipedia.org/wiki/UIMA> (cited on page 699).
- [706] Wikipedia. *Apache tomcat --- Wikipedia, The Free Encyclopedia*. [Accessed: 02-24-2017]. 2010. URL: [https://en.wikipedia.org/wiki/Apache\\_Tomcat](https://en.wikipedia.org/wiki/Apache_Tomcat) (cited on page 752).
- [707] Wikipedia. *EclipseLink --- Wikipedia, The Free Encyclopedia*. [Accessed: 02-06-2017]. 2010. URL: <https://en.wikipedia.org/wiki/EclipseLink> (cited on page 698).
- [708] Wikipedia. *Torch (Machine Learning) --- Wikipedia, The Free Encyclopedia*. [Accessed: 02-06-2017]. 2014. URL: [https://en.wikipedia.org/wiki/Torch\\_\(machine\\_learning\)](https://en.wikipedia.org/wiki/Torch_(machine_learning)) (cited on page 658).
- [709] Wikipedia. *CloudControl*. Wiki Page. Feb. 2016. URL: <https://en.wikipedia.org/wiki/CloudControl> (cited on page 671).
- [710] Wikipedia. *Key-value database*. WebPage. Nov. 2016. URL: [https://en.wikipedia.org/wiki/Key-value\\_database](https://en.wikipedia.org/wiki/Key-value_database) (cited on page 696).
- [711] Wikipedia. *Rasdaman --- Wikipedia, The Free Encyclopedia*. [Accessed: 02-23-2017]. 2016. URL: <https://en.wikipedia.org/wiki/Rasdaman> (cited on page 703).
- [712] Wikipedia. *Spanner (Database)*. Web Page. accessed 2017-01-29. Oct. 2016. URL: [https://en.wikipedia.org/wiki/Spanner\\_\(database\)](https://en.wikipedia.org/wiki/Spanner_(database)) (cited on page 710).
- [713] Wikipedia. *Amazon Redshift*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Amazon\\_Redshift](https://en.wikipedia.org/wiki/Amazon_Redshift) (cited on page 677).
- [714] Wikipedia. *Ansible (software)*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Ansible\\_\(software\)](https://en.wikipedia.org/wiki/Ansible_(software)) (cited on page 733).
- [715] Wikipedia. *Apache CloudStack*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Apache\\_CloudStack](https://en.wikipedia.org/wiki/Apache_CloudStack) (cited on page 744).
- [716] Wikipedia. *Apache Flink*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Apache\\_Flink](https://en.wikipedia.org/wiki/Apache_Flink) (cited on page 685).
- [717] Wikipedia. *Apache Hadoop*. Web Page. accessed 2017-03-18. Mar. 2017. URL: <https://en.wikipedia.org/wiki/Apache%5CHadoop> (cited on page 684).

- [718] Wikipedia. *Apache Phoenix* --- *Wikipedia, The Free Encyclopedia*. Web page. Online; accessed 25-jan-2017. Jan. 2017. URL: [https://en.m.wikipedia.org/wiki/Apache%5C\\_Phoenix](https://en.m.wikipedia.org/wiki/Apache%5C_Phoenix) (cited on page 674).
- [719] Wikipedia. *Bigtable*. Web Page. accessed 2017-01-29. Jan. 2017. URL: <https://en.wikipedia.org/wiki/Bigtable> (cited on page 710).
- [720] Wikipedia. *Chef (Software)*. Web Page. accessed 2017-01-26. Jan. 2017. URL: [https://en.wikipedia.org/wiki/Chef\\_\(software\)](https://en.wikipedia.org/wiki/Chef_(software)) (cited on page 737).
- [721] Wikipedia. *Cloud computing* --- *Wikipedia, The Free Encyclopedia*. web page. Online; accessed 21-jan-2017. Jan. 2017. URL: [https://en.wikipedia.org/wiki/Cloud\\_computing](https://en.wikipedia.org/wiki/Cloud_computing) (cited on page 668).
- [722] Wikipedia. *CUDA*. Web Page. Online; accessed 25-Feb-2017. Feb. 2017. URL: <https://en.wikipedia.org/wiki/CUDA> (cited on page 753).
- [723] Wikipedia. *Data Analytics Acceleration Library* --- *Wikipedia, The Free Encyclopedia*. [Accessed: 02-23-2017]. 2017. URL: [https://en.wikipedia.org/wiki/Data\\_Analytics\\_Acceleration\\_Library](https://en.wikipedia.org/wiki/Data_Analytics_Acceleration_Library) (cited on page 658).
- [724] Wikipedia. *Deeplearning4j*. Web Page. Accessed: 2017-02-11. Feb. 2017. URL: <https://en.wikipedia.org/wiki/Deeplearning4j> (cited on page 659).
- [725] Wikipedia. *Erlang (programming language)* --- *Wikipedia, The Free Encyclopedia*. Web page. Online; accessed: 29-jan-2017. Jan. 2017. URL: [https://en.wikipedia.org/wiki/Erlang\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Erlang_(programming_language)) (cited on page 709).
- [726] Wikipedia. *Fiddler Software*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Fiddler\\_\(software\)](https://en.wikipedia.org/wiki/Fiddler_(software)) (cited on page 752).
- [727] Wikipedia. *Graph Database*. Web Page. accessed 2017-01-30. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Graph\\_database](https://en.wikipedia.org/wiki/Graph_database) (cited on page 712).
- [728] Wikipedia. *Hazelcast*. Web Page. accessed 2017-01-29. Jan. 2017. URL: <https://en.wikipedia.org/wiki/Hazelcast> (cited on page 696).
- [729] Wikipedia. *Hierarchical Data Format* --- *Wikipedia, The Free Encyclopedia*. Web Page. Online; accessed: 2017-1-28. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Hierarchical\\_Data\\_Format](https://en.wikipedia.org/wiki/Hierarchical_Data_Format) (cited on page 717).
- [730] Wikipedia. *Hyper-V*. Web Page. Accessed: 2017-02-23. Feb. 2017. URL: <https://en.wikipedia.org/wiki/Hyper-V> (cited on page 741).
- [731] Wikipedia. *Hypervisor*. WebPage. Feb. 2017. URL: <https://en.wikipedia.org/wiki/Hypervisor> (cited on page 741).
- [732] Wikipedia. *IBM Cloudant*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: <https://en.wikipedia.org/wiki/Cloudant> (cited on page 710).
- [733] Wikipedia. *IBM General Parallel File System*. Web Page. accessed 2017-01-29. Jan. 2017. URL: [https://en.wikipedia.org/wiki/IBM\\_General\\_Parallel\\_File\\_System](https://en.wikipedia.org/wiki/IBM_General_Parallel_File_System) (cited on page 727).
- [734] Wikipedia. *MQTT* --- *Wikipedia, The Free Encyclopedia*. [Online; accessed 6-November-2017]. Nov. 2017. URL: <https://en.wikipedia.org/w/index.php?title=MQTT&oldid=808683219> (cited on page 440).
- [735] Wikipedia. *QEMU*. WebPage. Feb. 2017. URL: <https://en.wikipedia.org/wiki/QEMU> (cited on page 741).
- [736] Wikipedia. *Relational database*. WebPage. Jan. 2017. URL: [https://en.wikipedia.org/wiki/Relational\\_database](https://en.wikipedia.org/wiki/Relational_database) (cited on page 696).
- [737] Wikipedia. *Snort Software*. Web Page. Accessed: 2017-02-12. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Snort\\_\(software\)](https://en.wikipedia.org/wiki/Snort_(software)) (cited on page 751).



- [738] Wikipedia. *Storm (event processor)* --- Wikipedia, *The Free Encyclopedia*. [Online; accessed 6-November-2017]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Storm\\_\(event\\_processor\)&oldid=808771136](https://en.wikipedia.org/w/index.php?title=Storm_(event_processor)&oldid=808771136) (cited on page 442).
- [739] Wikipedia. *Swift (programming language)*. Web Page. Accessed: 2017-02-23. Feb. 2017. URL: [https://en.wikipedia.org/wiki/Swift\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Swift_(programming_language)) (cited on page 644).
- [740] Wikipedia. *Fusion Table Support*. Web Page. Last updated in 1 November 2016. URL: <https://support.google.com/fusiontables/answer/171181?hl=en> (cited on page 662).
- [741] Wikipedia. *LinkedIn*. Web page. accessed 17-March-2017. This page was last modified on 18 March 2017, at 22:27. URL: <https://cloud.google.com/ml-engine/docs/concepts/technical-overview> (cited on page 682).
- [742] Brandon Wiley. “Distributed Hash TTable, Part 1”. In: *Linux Journal* 114 (Oct. 2003). From issue number 114. URL: <http://www.linuxjournal.com/article/6797?page=0,0> (cited on pages 706, 707).
- [743] Duncan C.E Winn. *Cloud Foundry-The Cloud Native Platform*. O’Reilly Media, 2016, page 70. ISBN: 978-4919-6573-3. URL: <https://books.google.com/books?id=XP2wDAAAQBAJ&printsec=frontcover&dq=Cloud+foundry> (cited on page 668).
- [744] Alex Woodie. *Spark Gets New Machine Learning Framework: KeystoneML*. URL: <https://www.datanami.com/2015/05/26/spark-gets-new-machine-learning-framework-keystoneml/> (visited on 02/27/2017) (cited on page 652).
- [745] Alex Woodie. *Apache Beam’s Ambitious Goal: Unify Big Data Development*. Web page. accessed 25-feb-2017. Apr. 2016. URL: <https://www.datanami.com/2016/04/22/apache-beam-emerges-ambitious-goal-unify-big-data-development/> (cited on page 753).
- [746] Apache Software Foundation. *Apache Ranger*. Web Page. Online; accessed 9-Mar-2017. URL: <http://ranger.apache.org/> (cited on page 757).
- [747] CASK. *CASK - The First Unified Integration Platform for Big Data*. Web Page. Online; accessed 18-Feb-2017. URL: <http://cask.co/products/cdap/> (cited on page 753).
- [748] CASK. *Getting Started Developing with CDAP*. Web Page. Online; accessed 18-Feb-2017. URL: <http://docs.cask.co/cdap/current/en/developers-manual/getting-started/index.html> (cited on page 754).
- [749] *Open Source Data Turbine Initiative*. Web Page. Accessed: 2017-02-26. URL: <http://dataturbine.org/> (cited on page 684).
- [750] Jeff Lindsay. *Gitreceive*. Web Page. Accessed: 2017-02-13. Feb. 2016. URL: <https://github.com/progrium/gitreceive> (cited on page 738).
- [751] *High Performance ParalleX (HPX-5)*. Web Page. Accessed: 2017-1-17. URL: <https://hpx.crest.iu.edu/> (cited on page 689).
- [752] *HPX-5: user guide*. Accessed: 2017-1-17. HPX-5. URL: [https://hpx.crest.iu.edu/users\\_guide](https://hpx.crest.iu.edu/users_guide) (cited on page 689).
- [753] Open Grid Forum. *Open Cloud Computing Interface*. Web Page. Accessed: 2017-1-17. URL: <http://occi-wg.org/> (cited on page 731).
- [754] Ralf Nyren et al. *Open Cloud Computing Interface Core*. OGF Published Document GWD-R-P.221. Accessed: 2017-1-17. Open Grid Forum, P.O. Box 1738, Muncie IN 47308, USA: Global Grid Forum, Sept. 2016. URL: <https://www.ogf.org/documents/GFD.221.pdf> (cited on page 731).
- [755] Ralf Nyren, Andy Edmonds, and Thijs Metsch atnd Boris Parak. *Open Cloud Computing Interface - HTTP Protocol*. OGF Published Document GWD-R-P.223. Accessed: 2017-1-

17. Open Grid Forum, P.O. Box 1738, Muncie IN 47308, USA: Global Grid Forum, Sept. 2016. URL: <https://www.ogf.org/documents/GFD.223.pdf> (cited on page 731).
- [756] Ralf Nyren et al. *Open Cloud Computing Interface -JSON Rendering*. OGF Published Document GWD-R-P.226. Accessed: 2017-1-17. Open Grid Forum, P.O. Box 1738, Muncie IN 47308, USA: Global Grid Forum, Sept. 2016. URL: <https://www.ogf.org/documents/GFD.226.pdf> (cited on page 731).
- [757] Michel Drescher, Boris Parak, and David Wallom. *OCCI Compute Resource Templates Profile*. OGF Published Document GWD-R-P.222. Accessed: 2017-1-17. Open Grid Forum, P.O. Box 1738, Muncie IN 47308, USA: Global Grid Forum, Apr. 2015. URL: <https://www.ogf.org/documents/GFD.222.pdf> (cited on page 731).
- [758] Andy Edmonds and Thijs Metsch. *Open Cloud Computing Interface - Text Rendering*. OGF Published Document GWD-R-P.229. Accessed: 2017-1-17. Open Grid Forum, P.O. Box 1738, Muncie IN 47308, USA: Global Grid Forum, Sept. 2016. URL: <https://www.ogf.org/documents/GFD.229.pdf> (cited on page 731).
- [759] Hortonworks. *Apache Ranger - Overview*. Web Page. Online; accessed 9-Mar-2017. URL: [https://hortonworks.com/apache/ranger/#section\\_2](https://hortonworks.com/apache/ranger/#section_2) (cited on page 757).
- [760] Syed Mahmood and Srikanth Venkat. *FOR YOUR EYES ONLY: DYNAMIC COLUMN MASKING & ROW-LEVEL FILTERING IN HDP2.5*. Web Page. Online; accessed 9-Mar-2017. Sept. 2016. URL: <https://hortonworks.com/blog/eyes-dynamic-column-masking-row-level-filtering-hdp2-5/> (cited on page 757).
- [761] SAP. *What is SAP HANA*. Web Page. Accessed: 2017-1-17. URL: <http://www.sap.com/product/technology-platform/hana.html> (cited on page 675).
- [762] xcat. *Extreme cloud/cluster administration toolkit*. Webpage. 2015. URL: <http://xcat-docs.readthedocs.io/en/stable/> (cited on page 734).
- [763] *Xen - Wikipedia*. Web Page. Accessed: 2017-02-04. URL: <https://en.wikipedia.org/wiki/Xen> (cited on page 740).
- [764] *Xen Project Overview*. Web Page. Accessed: 2017-02-04. URL: [https://wiki.xenproject.org/wiki/Xen\\_Project\\_Software\\_Overview](https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview) (cited on page 740).
- [765] *Xen Feature List*. Web Page. Accessed: 2017-02-04. URL: [https://wiki.xenproject.org/wiki/Xen\\_Project\\_4.7\\_Feature\\_List](https://wiki.xenproject.org/wiki/Xen_Project_4.7_Feature_List) (cited on page 740).
- [766] Reynold S. Xin et al. “GraphX: A Resilient Distributed Graph System on Spark”. In: *First International Workshop on Graph Data Management Experiences and Systems*. GRADES '13. New York, New York: ACM, 2013, 2:1–2:6. ISBN: 978-1-4503-2188-4. DOI: 10.1145/2484425.2484427. URL: <http://doi.acm.org/10.1145/2484425.2484427> (cited on page 660).
- [767] T. Xu, K. Sato, and S. Matsuoka. “CloudBB: Scalable I/O Accelerator for Shared Cloud Storage”. In: *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*. Institute of Electrical and Electronics Engineers (IEEE), Dec. 2016, pages 509–518. DOI: 10.1109/ICPADS.2016.0074 (cited on page 727).
- [768] Shelhamer Yangqing Jia Evan. *Caffe | Deep Learning Framework*. Web Page. Accessed: 02-06-2017. URL: <http://caffe.berkeleyvision.org/> (cited on page 658).
- [769] Edward J. Yoon. *MRQL - a SQL on Hadoop Miracle*. Web Page. accessed 2017-01-29. Jan. 2017. URL: <http://www.hadoopsphere.com/2013/04/mrql-sql-on-hadoop-miracle.html> (cited on page 675).
- [770] Apache Zeppelin. *Apache Zeppelin 0.7.0 Documentation*. Web Page. Accessed: 2017-02-27. Feb. 2017. URL: <https://zeppelin.apache.org/docs/0.7.0/> (cited on page 752).
- [771] Bingjing Zhang et al. “Applying twister to scientific applications”. In: *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE. 2010, pages 25–32 (cited on page 685).

- [772] Xianbo Zhang et al. ‘‘Hptfs: A high performance tape file system’’. In: *Proceedings of 14th NASA Goddard/23rd IEEE conference on Mass Storage System and Technologies*. Ieee Computer Society (September 1995), 2006. DOI: [10.1.1.184.1133](https://doi.org/10.1.1.184.1133). URL: [https://www.dtc.umn.edu/publications/reports/2006\\_11.pdf](https://www.dtc.umn.edu/publications/reports/2006_11.pdf) (cited on page 727).
- [773] Jianlong Zhong and Bingsheng He. ‘‘Medusa: Simplified Graph Processing on GPUs’’. In: *IEEE Transactions on Parallel and Distributed Systems* 25.6 (Apr. 2013), pages 1–11. URL: [http://pdcc.ntu.edu.sg/xtra/paper/2013ago/Medusa\\_TPDS13.pdf](http://pdcc.ntu.edu.sg/xtra/paper/2013ago/Medusa_TPDS13.pdf) (cited on pages 687, 688).
- [774] *Zookeeper - Overview*. Web Page. Accessed: 2017-01-23. URL: <https://zookeeper.apache.org/doc/trunk/zookeeperOver.html> (cited on page 750).
- [775] *Zookeeper - Wikipedia*. Web Page. Accessed: 2017-01-23. URL: [https://en.wikipedia.org/wiki/Apache\\_ZooKeeper](https://en.wikipedia.org/wiki/Apache_ZooKeeper) (cited on page 750).
- [776] *IBM - What is Zookeeper*. Web Page. Accessed: 2017-01-23. URL: <http://www-01.ibm.com/software/data/infosphere/hadoop/zookeeper/> (cited on page 750).



# Index

- .bashrc, [213](#)
- L<sup>A</sup>T<sub>E</sub>X
  - chktex, [174](#)
  - lacheck, [173](#)
- Assignments
  - E222, [102](#)
  - E516, [104](#)
  - E616, [104](#)
  - I524, [104](#)
- Auditing, [120](#)
- autopep8, [324](#)
- Bayes, [621](#)
- biber, [177](#)
- Bibliography, [177](#)
- Bibtex
  - Article, [181](#), [187](#)
  - Blog, [188](#)
  - book, [189](#)
  - Field
    - Accessed, [180](#)
    - Author, [180](#)
    - Howpublished, [180](#)
    - Key, [180](#)
    - Label, [179](#)
    - Misc, [179](#)
    - Month, [180](#)
    - Owner, [180](#)
  - InProceedings, [183](#), [186](#)
  - MSWord, [192](#)
  - Proceedings, [187](#)
  - Source code, [179](#)
  - TechReport, [186](#)
  - Web Page, [189](#)
  - Wikipedia Entry, [188](#)
- bibtex, [177](#)
- bibtex4word, [192](#)
- Big Data Applications, [595](#)
- Binomial Distribution, [621](#)
- Calendar, [115](#), [118](#)
- CANVAS, [115](#)
- Central Limit Theorem, [620](#), [621](#)
- Chameleon, [247](#)
  - Bare Metal, [284](#)
  - Openstack, [506](#)
- chktex, [174](#)
- chocolatey, [220](#)
- Class Material, [117](#)
- Code
  - K-means
    - kmeans-extra.py, [870](#)
    - pagerank1.py, [871](#)
    - pagerank2.py, [871](#)
    - parallel-kmeans.py, [870](#)
    - ParallelKmeans, [871](#)
    - sample.csv, [870](#)
    - xmean.py, [869](#)
  - Physics



- HiggsClassI-Sloping.py, [617](#), [618](#)
- HiggsClassIII.py, [618](#)--[620](#)
- HiggsClassIIUniform.py, [618](#)
- Plotviz
  - clusterFinal-M3-C3Dating-ReClustered.pviz, [874](#)
  - ClusterFinal-M30-C28.pviz, [873](#)
  - DatingRatings-OriginalLabels.pviz, [873](#)
  - Fungi-LSU-3-15-to-3-26-zeroidx.pviz, [873](#)
- Contributing, [61](#), [62](#)
- Convention, [63](#)
  - Variables, [64](#)
- Dask, [336](#)
- Data
  - Formats, [549](#)
- Django
  - REST, [423](#)
- Docker, [452](#)
  - Flask REST API, [461](#)
  - Futuresystems, [459](#)
  - Install
    - OSX, [455](#)
    - Test, [457](#)
    - Ubuntu, [456](#)
    - Windows, [456](#)
  - Local, [455](#)
- Dockerhub, [497](#)
- e-Commerce, [611](#)
- Endnote, [192](#)
- Evegenie, [421](#)
- Event Counting, [619](#)
- Filtering, [614](#)
- Fox, Geoffrey C, [60](#)
- FutureSystems
  - Bravo, [242](#)
  - Delta, [243](#)
  - Echo, [243](#)
  - Juliet, [243](#)
  - PI Cluster, [244](#)
  - Romeo, [243](#)
  - Tango, [243](#)
  - Tempest, [244](#)
  - Victor, [244](#)
- Futuresystems
  - Docker, [459](#)
  - Kubernetes, [473](#)
- futuresystems, [241](#)
- Gaussian Distributions, [620](#)
- Generators, [620](#)
- Google News, [613](#)
- guizero, [324](#)
- Hadoop, [513](#)
- Harp, [537](#)
- Helath Informatics, [605](#)
- HID, [117](#)
- Higgs Particles, [617](#)
- Incomplete, [118](#)
- IU
  - Register, [120](#)
- JSON, [550](#)
- jupyter, [327](#)
- k-Nearest Neighbors, [615](#)
- Kaggle, [612](#)
- kivy, [324](#)
- Kuberneted
  - Deplyments, [470](#)
  - Services, [470](#)
- Kubernetes, [469](#)
  - Futuresystems, [473](#)
  - Minikube, [471](#)
- lacheck, [173](#)
- Latex
  - #, [160](#)
  - \$, [160](#)
  - %, [160](#)
  - \_, [160](#)
  - cycle, [171](#)
  - Elements, [160](#)
    - description, [161](#)
    - enumerate, [161](#)
    - highlight text, [160](#)
    - images, [162](#)
    - itemiz, [161](#)
    - labels, [164](#)
    - mathematics, [164](#)
    - sections, [160](#)
    - tables, [162](#)
  - installation, [157](#)
    - OSX, [158](#)
    - ubuntu, [158](#)
    - Windows, [158](#)
  - markdown, [170](#)
  - other documentation, [175](#)

- overleaf, [159](#)
- proceedings
  - acm, [165](#)
  - ieee, [165](#)
- Quicklatex, [159](#)
- Sharelatex, [158](#)
- slides, [167](#)
- LifeStyle, [611](#)
- Linux, [209](#)
- Makefile, [213](#)
  - Windows, [214](#)
- MapReduce, [513](#)
- Mendeley, [192](#)
- Monte Carlo, [619](#)
- Monte Carlo Method, [621](#)
- Nearest Neighbor, [614](#)
- Netflix, [613](#)
- NIST
  - Use Case, [595](#)
- Normal Distributions, [620](#)
- notebook.md, [117](#)
- Office Hours, [116](#)
- Paperpile, [193](#)
- Physics, [617](#)
- pip, [323](#)
- Plagiarism, [121](#)
- Pods, [470](#)
- Poisson Distribution, [621](#)
- Policies, [115](#)
- Projects, [919](#)
- putty, [219](#)
  - Futuresystems, [242](#)
- pyenv, [293](#)
- pypi, [325](#)
- Python, [291](#)
  - 2 and 3, [301](#)
  - boolean, [304](#)
  - classes, [313](#)
  - data types, [304](#)
  - date, [306](#)
  - dict, [311](#)
  - for, [308](#)
  - function, [313](#)
  - if, [307](#)
  - import, [305](#)
    - from, [305](#)
  - Install, [293](#)
  - interactive, [300](#)
  - keys, [312](#)
  - Libraries, [323](#)
  - list, [308](#)
  - numbers, [304](#)
  - pip, [323](#)
  - References, [292](#)
  - REPL, [300](#)
  - set, [310](#)
  - statements, [303](#)
  - strings, [303](#)
  - time, [306](#)
  - variables, [303](#)
- Random Variables, [619](#)
- Recommender Systems, [612](#)
- Responses, [423](#)
- REST, [407](#), [408](#), [410](#)
  - Eve, [410](#)
  - Extensions, [420](#)
  - Evegenie, [421](#)
  - Flask RESTful, [408](#)
  - HATEOS, [419](#)
  - Overview, [407](#)
  - Swagger, [424](#)
- Seeds, [620](#)
- Shell, [209](#)
  - Variables, [64](#)
- SKA, [621](#)
- Slides
  - About
    - Test slides (10), [63](#)
  - Cloud
    - Analysis Algorithms (Page 35), [555](#), [894](#)
    - Analysis Algorithms - pptx (Page 35), [555](#), [894](#)
    - Apache Data Analysis OpenStack (Page 1), [901](#)
    - Apache Data Analysis OpenStack - pptx (Page 1), [901](#)
    - BigTable (Page 28), [552](#), [892](#)
    - BigTable - pptx (Page 28), [553](#), [892](#)
    - BLAST Parallelization (Page 13), [904](#)
    - BLAST Parallelization - pptx (Page 13), [904](#)
    - Challenges (Page 42), [884](#)
    - Checklists and Challenges (Page 11), [887](#)
    - Clouds in the Workplace (Page 1), [887](#)

- Clusters and Resource Management (Page 66), [886](#)
- Clusters and Resource Management - pptx (Page 66), [887](#)
- Coding and Iterative Alternatives (Page 43), [910](#)
- Coding and Iterative Alternatives - pptx (Page 43), [911](#)
- CPU, Memory & I/O Devices (Page 58), [886](#)
- CPU, Memory & I/O Devices - pptx (Page 58), [886](#)
- Cultivating Clouds (Page 15), [888](#)
- Cultivating Clouds - Conclusions (Page 1), [888](#)
- Data Center Automation (Page 74), [887](#)
- Data Center Automation - pptx (Page 74), [887](#)
- Data Center Model (Page 9), [883](#)
- Data Center Setup (Page 16), [888](#)
- Data Intensive Sciences (Page 19), [884](#)
- Data Locality (Page 10), [904](#)
- Data Locality - pptx (Page 10), [904](#)
- Discussions and ParallelThinking (Page 10), [908](#)
- Discussions and ParallelThinking - pptx (Page 10), [908](#)
- Everyday Data (Page 4), [915](#)
- Fault Tolerance (Page 36), [902](#)
- Fault Tolerance - pptx (Page 36), [902](#)
- Faults & Frameworks (Page 9), [916](#)
- Google Architecture (Page 6), [897](#)
- Google Architecture - pptx (Page 6), [897](#)
- Google Components (Page 1), [897](#)
- Google Components - pptx (Page 1), [897](#)
- Google History (Page 14), [897](#)
- Google History - pptx (Page 14), [898](#)
- Google Search Engine 1 (Page 15), [907](#)
- Google Search Engine 1 - pptx (Page 15), [907](#)
- Google Search Engine 2 (Page 21), [908](#)
- Google Search Engine 2 - pptx (Page 21), [908](#)
- Growth of Virtual Machines (Page 28), [885](#)
- Growth of Virtual Machines - pptx (Page 28), [885](#)
- Hadoop Extensions (Page 50), [909](#)
- Hadoop Framework (Page 15), [902](#)
- Hadoop Framework - pptx (Page 15), [902](#)
- Hadoop PageRank (Page 1), [908](#)
- Hadoop PageRank - pptx (Page 1), [908](#)
- Hadoop Tasks (Page 24), [902](#)
- Hadoop Tasks - pptx (Page 24), [902](#)
- Hadoop WordCount on VMs (Page 17), [902](#)
- Hadoop WordCount on VMs - pptx (Page 17), [902](#)
- HBase (Page 44), [553](#), [892](#)
- HBase - pptx (Page 44), [553](#), [892](#)
- HBase Coding (Page 60), [553](#), [892](#)
- HBase Coding - pptx (Page 60), [553](#), [892](#)
- How Hadoop Runs on a MapReduceJob (Page 8), [903](#)
- How Hadoop Runs on a MapReduceJob - pptx (Page 8), [903](#)
- IaaS, PaaS and SaaS (Page 25), [884](#)
- Implementation Levels (Page 41), [886](#)
- Implementation Levels - pptx (Page 41), [886](#)
- Indexamples (Page 15), [554](#), [893](#)
- Indexamples - pptx (Page 15), [554](#), [893](#)
- Indexing 101 (Page 20), [554](#), [893](#)
- Indexing 101 - pptx (Page 20), [554](#), [893](#)
- Indexing Applications (Page 1), [553](#), [892](#)
- Indexing Applications - pptx (Page 1), [554](#), [892](#)
- Introduction - (retired) (Page 1), [883](#)
- Introduction - Part A (7 Slides), [400](#)
- Introduction to BLAST (Page 1), [903](#)
- Introduction to BLAST - pptx (Page 1), [903](#)
- Iterative MapReduce Models (Page 1), [909](#)
- Iterative MapReduce Models - pptx (Page 1), [909](#)
- Literature Review (Page 16), [903](#)
- Literature Review - pptx (Page 16), [903](#)
- MapReduce (Page 6), [901](#)
- MapReduce - pptx (Page 6), [901](#)
- MapReduce Model Comparison (Page 24), [910](#)
- MapReduce Model Comparison - pptx (Page 24), [910](#)
- MapReduce Refresher (Page 1), [907](#)



- NoSQL Characteristics (Page 11), [552](#), [891](#)
- NoSQL Characteristics - pptx (Page 11), [552](#), [891](#)
- Optimal Data Locality (Page 17), [905](#)
- Optimal Data Locality - pptx (Page 17), [905](#)
- Parallel Processes (Page 4), [909](#)
- Parallel Processes - pptx (Page 4), [909](#)
- Part B - Defining Clouds I (13 Slides), [400](#)
- Part C - Defining Clouds II (11 Slides), [401](#)
- Part D - Defining Clouds III (9 Slides), [401](#)
- Part E - Virtualization (8 Slides), [401](#)
- Part F - Technology Hypecycle I (11 Slides), [401](#)
- Part G - Technology Hypecycle II (15 Slides), [401](#)
- Part H - IaaS I (12 Slides), [402](#)
- Part I - IaaS II (11 Slides), [402](#)
- Part J - Cloud Software (15 Slides), [402](#)
- Part K - Applications I (16 Slides), [402](#)
- Part L - Applications II (11 Slides), [403](#)
- Part M - Applications III (14 Slides), [403](#)
- Part N - Parallelism (15 Slides), [403](#)
- Part O - Storage (10 Slides), [403](#)
- Part P - HPC in the Clou (8 Slides), [403](#)
- Part Q - Analytics and Simulation (10 Slides), [403](#)
- Part R - Jobs (6 Slides), [404](#)
- Part S - The Future (6 Slides), [404](#)
- Part T - Security (13 Slides), [404](#)
- Part U - Fault Tolerance (5 Slides), [404](#)
- Programming on a Compute Cluster (Page 1), [903](#)
- Programming on a Compute Cluster - pptx (Page 1), [903](#)
- RDBMS vs. NoSQL (Page 1), [551](#), [891](#)
- RDBMS vs. NoSQL - pptx (Page 1), [551](#), [891](#)
- Related Work (Page 11), [554](#), [893](#)
- Related Work - pptx (Page 11), [554](#), [893](#)
- Resource Utilization and Speculative Execution (Page 46), [905](#)
- Resource Utilization and Speculative Execution - pptx (Page 46), [905](#)
- SIMD vs MIMD;SPMD vs MPMD (Page 1), [904](#)
- SIMD vs MIMD;SPMD vs MPMD - pptx (Page 1), [904](#)
- Social Media Searches (Page 28), [555](#), [894](#)
- Social Media Searches - pptx (Page 28), [555](#), [894](#)
- Spouts to Bolts (Page 15), [916](#)
- Static and Variable Data (Page 10), [910](#)
- Static and Variable Data - pptx (Page 10), [910](#)
- Streaming the Data Ocean (Page 6), [915](#)
- Streams of Events (Page 1), [915](#)
- Student Work 1 (Page 7), [61](#)
- Student Work 1 (pptx) (Page 7), [61](#)
- Student Work 2 (Page 12), [61](#)
- Student Work 2 (pptx) (Page 12), [61](#)
- Task Granularity (Page 29), [905](#)
- Task Granularity - pptx (Page 29), [905](#)
- Tools and Mechanisms (Page 47), [886](#)
- Tools and Mechanisms - pptx (Page 47), [886](#)
- Twister K-means (Page 34), [910](#)
- Twister K-means - pptx (Page 34), [910](#)
- Health
  - Health Informatics (131), [605](#)
  - Proteomics and Information Visualization (131), [608](#)
- i524
  - Access Patterns, Data Access Patterns and Introduction to using HPC-ABDS (Pages ??? ), [639](#)
  - Course Introduction (Pages 39), [639](#)
  - Overview (Pages 26), [639](#)
- Lifestyle
  - 18 (Filtering), [614](#)
  - 45 (Recommender), [612](#)
  - 49 (Recommender), [613](#)
- Physics
  - Higgs (20), [617](#)
  - Higgs (39), [619](#)
  - Higgs II (29), [618](#)
  - Higgs III (44), [620](#)
- Plotviz
  - Plotvisz (34), [873](#)
- QEMU-KVM
  - QEMU KVM (50), [452](#), [505](#)
- Radar

- Radar (58), [622](#)
- Sensor
  - Sensor I (31), [625](#)
  - Sensor II (44), [625](#)
- Sport
  - 44 (Sports III), [631](#)
  - Overview (40), [629](#)
  - Sporta II (41), [630](#)
- Usecases
  - 100 (51), [598](#)
  - 43 (Features), [600](#)
  - 45 (Overview), [595](#)
- Web
  - Text Mining (33), [635](#)
  - Web Search and Text Mining (56), [634](#)
- Spark, [519](#)
- Sports, [629](#)
- Square Kilometer Array, [621](#)
- ssh, [215](#)
  - Futuresystems, [222](#), [242](#)
- ssh add, [217](#)
- ssh config, [217](#)
- ssh port forwarding, [217](#)
- ssh-keygen, [215](#)
- Statistics, [620](#)
- Swagger, [424](#)
- Terminal, [209](#)
- Twister, [539](#)
- Variables, [64](#)
  - Shell, [64](#)
- Vector Space, [614](#)
- Video
  - About
    - Fix Typo Video (01:18), [62](#)
    - Test Video (25:36), [63](#)
  - Bibliography
    - jabref (1:41), [178](#)
  - Class Management
    - Class Management - Overview (39:54), [79](#), [88](#)
  - Cloud
    - 4:30 (HBase Coding), [553](#), [892](#)
    - Analysis Algorithms (6:57), [555](#), [894](#)
    - Apache Data Analysis OpenStack (12:01), [901](#)
    - BigTable (6:55), [552](#), [892](#)
    - BLAST Parallelization (4:44), [904](#)
    - Challenges (5:27), [884](#)
    - Checklists and Challenges (9:08), [887](#)
    - Clouds in the Workplace (7:13), [887](#)
    - Clusters and Resource Management (5:07), [886](#)
    - Coding and Iterative Alternatives (5:14), [910](#)
    - CPU, Memory & I/O Devices (6:41), [886](#)
    - Cultivating Clouds (5:10), [888](#)
    - Data Center Automation (3:30), [887](#)
    - Data Center Model (8:08), [883](#)
    - Data Center Setup (7:49), [888](#)
    - Data Intensive Sciences (2:44), [884](#)
    - Data Locality (8:36), [904](#)
    - Discussions and ParallelThinking (11:12), [908](#)
    - Emacs org-mode (18:04), [197](#)
    - Everday Data (9:31), [915](#)
    - Fault Tolerance (2:45), [902](#)
    - Faults & Frameworks (7:46), [916](#)
    - Google Architecture (8:40), [897](#)
    - Google Components (7:02), [897](#)
    - Google History (10:36), [897](#)
    - Google Search Engine 1 (8:04), [907](#)
    - Google Search Engine 2 (8:32), [908](#)
    - Growth of Virtual Machines (10:16), [885](#)
    - Hadoop Extensions (5:37), [909](#)
    - Hadoop Framework (8:32), [902](#)
    - Hadoop PageRank (7:58), [908](#)
    - Hadoop Tasks (11:01), [902](#)
    - Hadoop WordCount on VMs (7:30), [902](#)
    - HBase (7:37), [553](#), [892](#)
    - How Hadoop Runs on a MapReduceJob (9:25), [903](#)
    - IaaS/PaaS/SaaS (10:17), [884](#)
    - Implementation Levels (7:57), [886](#)
    - Indexamples (8:35), [554](#), [893](#)
    - Indexing 101 (9:53), [554](#), [893](#)
    - Indexing Applications (9:33), [553](#), [892](#)
    - Introduction - (retired) (8:31), [883](#)
    - Introduction - Part A (14:48), [400](#)
    - Introduction to BLAST (8:27), [903](#)
    - Iterative MapReduce Models (6:46), [909](#)
    - Literature Review (9:43), [903](#)
    - MapReduce (9:07), [901](#)
    - MapReduce Model Comparison (6:56), [910](#)
    - MapReduce Refresher (9:00), [907](#)
    - NoSQL Characteristics (10:31), [552](#), [891](#)

- Optimal Data Locality (4:17), [905](#)
- Parallel Processes (9:44), [909](#)
- Part B - Defining Clouds I (20:22), [400](#)
- Part C - Defining Clouds II (20:45), [401](#)
- Part D - Defining Clouds III (9:08), [401](#)
- Part E - Virtualization (11:21), [401](#)
- Part F - Technology Hypecycle I (13:41), [401](#)
- Part G - Technology Hypecycle II (16:05), [401](#)
- Part H - IaaS I (13:22), [402](#)
- Part I - IaaS II (13:13), [402](#)
- Part J - Cloud Software (37:56), [402](#)
- Part K - Applications I (11:58), [402](#)
- Part L - Applications II (13:03), [403](#)
- Part M - Applications III (24:12), [403](#)
- Part N - Parallelism (35:46), [403](#)
- Part O - Storage (19:22), [403](#)
- Part P - HPC in the Clou (19:29), [403](#)
- Part Q - Analytics and Simulation (16:19), [403](#)
- Part R - Jobs (4:52), [404](#)
- Part S - The Future (19:46), [404](#)
- Part T - Security (11:29), [404](#)
- Part U - Fault Tolerance (9:10), [404](#)
- Programming on a Compute Cluster (6:01), [903](#)
- RDBMS vs. NoSQL (9:22), [551](#), [891](#)
- Related Work (5:56), [554](#), [893](#)
- Resource Utilization and Speculative Execution (3:52), [905](#)
- SIMD vs MIMD;SPMD vs MPMD (9:42), [904](#)
- Social Media Searches (6:19), [555](#), [894](#)
- Spouts to Bolts (8:42), [916](#)
- Static and Variable Data (11:01), [910](#)
- Streaming the Data Ocean (9:38), [915](#)
- Streams of Events (10:44), [915](#)
- Student Work 1 (8:48), [61](#)
- Student Work 2 (10:03), [61](#)
- Task Granularity (9:51), [905](#)
- Tools and Mechanisms (7:32), [886](#)
- Twister K-means (7:28), [910](#)
- Container
  - Container A (11:01), [451](#)
  - Container B (15:08), [451](#)
  - Container C (40:09), [451](#)
  - Container D (50:14), [452](#)
- Git
  - Branch (2:25), [227](#)
  - Checkout (3:11), [227](#)
  - Configuration (2:47), [228](#)
  - Fork (1:41), [226](#)
  - GUI (3:47), [227](#)
  - Merge (3:11), [227](#)
  - Pull Request (4:26), [227](#)
  - Rebase (4:20), [226](#)
  - Windows (1:25), [228](#)
- Github
  - Issues (8:29), [232](#)
- Hadoop
  - Hadoop A (19:02), [513](#)
  - Hadoop B (13:19), [513](#)
  - Hadoop C (12:57), [514](#)
  - Hadoop D (15:14), [514](#)
  - Hadoop E (14:55), [514](#)
  - Hadoop E (15:14), [514](#)
- Health
  - Big Data and Health (10:02), [605](#)
  - Clouds and Health (4:35), [606](#)
  - EU Report on Redesigning health in Europe for 2020 (5:00), [607](#)
  - Extrapolating to 2032 (15:13), [607](#)
  - Genomics, Proteomics and Information Visualization (6:56), [607](#)
  - Genomics, Proteomics and Information Visualization I (10:33), [607](#)
  - Genomics, Proteomics and Information Visualization: II (7:41), [608](#)
  - McKinsey Report (14:53), [606](#)
  - Medical Big Data in the Clouds (15:02), [606](#)
  - Medicine and the Internet of Things (8:17), [607](#)
  - Microsoft Report on Big Data in Health (2:26), [606](#)
  - Midical Image Big Data (6:33), [606](#)
  - Status of Healthcare Today (16:09), [606](#)
  - Telemedicine (8:21), [606](#)
- i524
  - A. Introduction to HPC-ABDS Software and Access Patterns (0:27:45), [639](#)
  - B. Science Examples (Data Access Patterns) (0:18:38), [639](#)
  - Basic Trends and Jobs (0:10:57), [639](#)
  - Big Data Patterns - Sources of Parallelism (0:23:51), [640](#)
  - Big Data Patterns - the Ogres and their

- Facets I (0:22:44), [640](#)
- C. Remaining General Access Patterns (11:26), [640](#)
- Clouds vs HPC - Data Intensive vs. Simulation Problems (0:20:26), [640](#)
- D. Summary HPC-ABDS Layers 1 - 6 (14:32), [640](#)
- E. Summary HPC-ABDS Layers 7 - 13 (30:52), [640](#)
- F. Summary HPC-ABDS Layers 14 - 17 (28:02), [640](#)
- Facets of the Big Data Ogres II (0:15:09), [640](#)
- First and Second Set of Features (0:18:26), [640](#)
- G. Summary HPC-ABDS Others (20:20), [640](#)
- Image Based Applications II (0:15:23), [640](#)
- Internet of Things Based Applications (0:25:25), [640](#)
- Introduction (0:13:59), [639](#)
- jabref (14:41), [145](#)
- Machine Learning Aspect of Second Feature Set and the Third Set (0:18:38), [640](#)
- More of Software Stack (0:24:00), [640](#)
- NIST Big Data Sub Groups (0:23:25), [640](#)
- NIST UseCases and Image Based Applications Examples I (0:25:20), [640](#)
- Other sources of use cases and Classical Databases/SQL Solutions (0:16:50), [640](#)
- Part 1 (11:29), [639](#)
- Part 2 (04:10), [639](#)
- Part 3 (12:41), [639](#)
- Real World Big Data (0:15:28), [639](#)
- ShareLaTeX (8:49), [145](#)
- SQL Solutions - Machine Learning Example - and MapReduce (0:18:49), [640](#)
- Lifestyle
  - Case Study of Recommender systems (3:21), [614](#)
  - Consumer Data Science (13:04), [613](#)
  - Examples of Recommender Systems (1:00), [612](#)
  - Examples of Recommender Systems (8:34), [613](#)
  - Item Based Filtering (11:18), [614](#)
  - k Nearest Neighbors and High Dimensional Spaces (10:03), [615](#)
  - k Nearest Neighbors and High Dimensional Spaces (7:16), [615](#)
  - Kaggle Competitions: (3:36), [612](#)
  - Netflix on Recommender Systems (14:20), [613](#)
  - Recap and Examples of Recommender Systems (5:48), [613](#)
  - Recap of Recommender Systems II (8:46), [614](#)
  - Recap of Recommender Systems III (10:48), [614](#)
  - Recommender Systems I (8:06), [612](#)
  - Recommender Systems Introduction (12:56), [612](#)
  - User-based nearest-neighbor collaborative filtering I (7:20), [614](#)
  - User-based nearest-neighbor collaborative filtering II (7:29), [614](#)
  - Vector Space Formulation of Recommender Systems new (9:06), [614](#)
- Physics
  - Accelerator Picture Gallery of Big Science (11:21), [618](#)
  - Accept-Reject (5:54), [621](#)
  - Binomial Distribution: (12:38), [621](#)
  - Central Limit Theorem (4:47), [621](#)
  - Change shape of background & num of Higgs Particles (7:01), [619](#)
  - Discovery of Higgs Particle (13:49), [618](#)
  - Event Counting (7:02), [619](#)
  - Gaussian Distributions (9:08), [620](#)
  - Generators and Seeds II (7:10), [621](#)
  - Higgs Particle Counting Errors (6:28), [621](#)
  - Interpretation of Probability (12:39), [621](#)
  - Looking for Higgs Particle and Counting Introduction II (7:38), [618](#)
  - Looking for Higgs Particle Experiments (9:29), [618](#)
  - Monte Carlo Method (2:23), [621](#)
  - Physics and Random Variables I (8:34), [620](#)
  - Physics and Random Variables II (5:50), [620](#)
  - Poisson Distribution (4:37), [621](#)

- Random variables and normal distributions (8:19), [619](#)
- Statistics of Events with Normal Distributions (11:25), [620](#)
- Using Statistics (14:02), [620](#)
- With Python examples of Signal plus Background (7:33), [619](#)
- Python
  - Advanced SSH (2:47), [242](#)
  - FutureGrid Introduction (11:50), [241](#)
  - Shell Access via SSH (2:34), [242](#)
  - ssh-key gen (4:06), [242](#)
  - Windows users (3:51), [242](#)
- Radar
  - Global Climate Change (2:51), [623](#)
  - Ice Sheet Science (1:00), [622](#)
  - Radar Informatics (3:31), [622](#)
  - Radio Informatics (3:35), [623](#)
  - Radio Overview (4:16), [623](#)
  - Remote Sensing (6:43), [622](#)
- REST
  - REST (36:02), [407](#)
  - Swagger (36:02), [426](#)
- Sensor
  - Earth/Environment/Polar Science data gathered by Sensors (4:58), [626](#)
  - Industrial Internet of Things (24:02), [626](#)
  - Internet of Things (12:36), [625](#)
  - Robotics and IoT Expectations (8:05), [626](#)
  - Sensor Clouds (4:40), [626](#)
  - Smart Grid (6:04), [627](#)
  - U-Korea (U=Ubiquitous) (2:49), [626](#)
  - Ubiquitous/Smart Cities (1:44), [626](#)
- Spark
  - Spark A (15:57), [519](#)
  - Spark B (12:17), [519](#)
  - Spark C (10:37), [520](#)
  - Spark D (26:18), [520](#)
- Sport
  - Basic Sabermetrics (26:53), [630](#)
  - Introduction and Sabermetrics (Baseball Informatics) Lesson (31:4), [629](#)
  - Other Video Data Gathering in Baseball (18:5), [630](#)
  - Pitcher Quality (10:02), [630](#)
  - PITCHf/X (10:39), [630](#)
  - Pitching Clustering (20:59), [630](#)
  - Soccer and the Olympics (8:28), [631](#)
  - Spatial Visualization in NFL and NBA (15:19), [631](#)
  - Tennis and Horse Racing (8:52), [631](#)
  - Wearables (22:2), [631](#)
  - Wins Above Replacement (30:43), [630](#)
- Systems
  - FutureGid (12:12), [241](#)
- Teaching Cloud and Big Data
  - Teaching Cloud and Big Data - Overview (39:54), [79](#), [88](#)
- Usecases
  - Architecture (10:05), [596](#)
  - Astronomy and Physics Use Cases (17:33), [599](#)
  - Commercial Use Cases (17:43), [598](#)
  - Database (SQL) Use Case Classification (11:13), [600](#)
  - Deep Learning and Social Networks Use Cases (14:19), [599](#)
  - Defense Use Cases (15:43), [598](#)
  - Energy Use Case (4:01), [599](#)
  - Environment, Earth and Polar Science Use Cases (25:29), [599](#)
  - Government Use Cases (17:43), [598](#)
  - Healthcare and Life Science Use Cases (30:11), [599](#)
  - Introduction (13:02), [596](#)
  - NoSQL Use Case Classification (11:20), [600](#)
  - Requirements (27:28), [598](#)
  - Research Ecosystem Use Cases (9:09), [599](#)
  - Security (9:51), [596](#)
  - Summary of Use Case Classification (23:39), [600](#)
  - Taxonomies (7:42), [596](#)
  - Technology (4:14), [597](#)
  - Use Case Classifications I (12:42), [600](#)
  - Use Case Classifications II (20:18), [600](#)
  - Use Case Classifications III (17:25), [601](#)
- Virtualbox
  - Video (4:46), [236](#)
  - Video (seconds), [236](#)
- Web
  - Boolean and Vector Space Model (6:17), [635](#)
  - Clustering and Topic Models (6:21), [636](#)
  - Fundamental Principals of Web Search (5:06), [634](#)

- Indices (5:44), [635](#)
- Information Retrieval (6:06), [634](#)
- Principles (9:30), [634](#)
- Realated Applications (17:24), [636](#)
- Search Engines (3:08), [635](#)
- Text Mining (9:56), [634](#)
- TF-IDF and Probabilistic Models (3:57),  
[635](#)
- Web Advertising and Search (9:02), [636](#)
- Web crawling and Document Preparation (4:55), [635](#)
- Web Search and Text Mining II (6:11),  
[635](#)
- Web Search History (5:48), [634](#)
- Writing
  - How to write a paper by Simon Peyton  
Jones (57:39), [145](#)
- virtualenv, [299](#)
- von Laszewski, Gregor, [60](#)
  
- Waitlist, [119](#)
- Writing, [149](#)
  - Checklist, [151](#)
  
- XML, [550](#)
  
- Yahoo, [613](#)
- YAML, [549](#)
  
- Zotero, [193](#)