HPC-ABDS High Performance Computing Enhanced Apache Big Data Stack

Geoffrey Fox[a], Judy Qiu[a], Shantenu Jha[b], Supun Kamburugamuve[a] and Andre Luckow[b]
[a] School of Informatics and Computing, Indiana University, Bloomington, IN 47408, USA
[b] RADICAL, Rutgers University, Piscataway, NJ 08854, USA

Abstract

We review the High Performance Computing Enhanced Apache Big Data Stack HPC-ABDS and summarize the capabilities in 21 identified architecture layers. These cover Message and Data Protocols, Distributed Coordination, Security & Privacy, Monitoring, Infrastructure Management, DevOps, Interoperability, File Systems, Cluster & Resource management, Data Transport, File management, NoSQL, SQL (NewSQL), Extraction Tools, Object-relational mapping, In-memory caching and databases, Inter-process Communication, Batch Programming model and Runtime, Stream Processing, High-level Programming, Application Hosting and PaaS, Libraries and Applications, Workflow and Orchestration. We summarize status of these layers focusing on issues of importance for data analytics. We highlight areas where HPC and ABDS have good opportunities for integration.

1. Introduction to HPC-ABDS

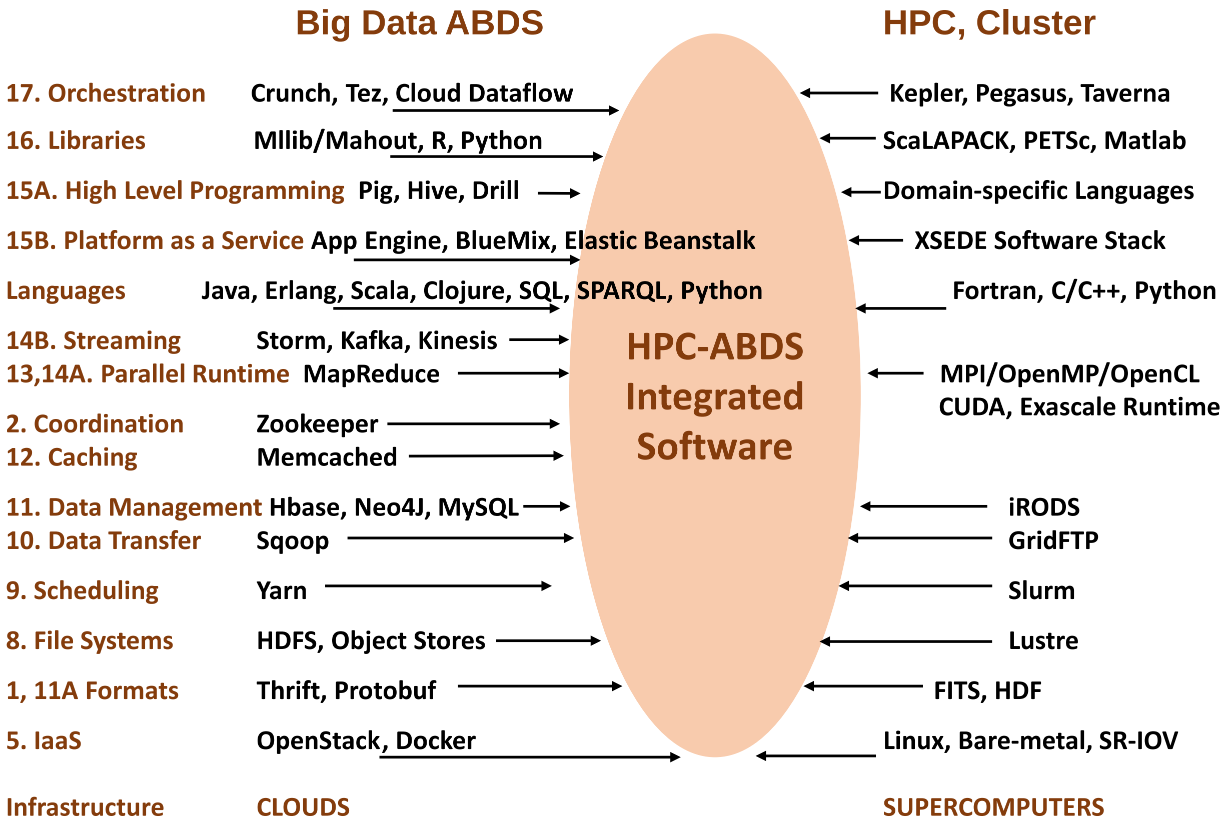

Figure 1: Comparison of typical Cloud and Supercomputer Software Layered Stacks

In previous work [1-3], we introduced the software stack HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) shown online [4] and in Figure 2. These were combined with an application analysis [5-7] and used to motivate an approach to high performance data analytics including identification of a benchmarking set [8, 9]. In this paper we focus on the stack of Figure 2 and describe its capabilities; inevitably this implies the paper has a review focus with original research correspondingly to the novel integration implied by the HPC-ABDS concept. Further details of the stack can be found in an online course [10] that includes a section with approximately one slide (and associated lecture video) for each entry in Figure 2. Links for all the technologies of Figure 2 can be found online [4, 10].

Figure 2: Kaleidoscope of (Apache) Big Data Stack (ABDS) and HPC Technologies

Cross-Cutting Functions	**17) Workflow-Orchestration:** ODE, ActiveBPEL, Airavata, Pegasus, Kepler, Swift, Taverna, Triana, Trident, BioKepler, Galaxy, IPython, Dryad, Naiad, Oozie, Tez, Google FlumeJava, Crunch, Cascading, Scalding, e-Science Central, Azure Data Factory, Google Cloud Dataflow, NiFi (NSA)
1) **Message and Data Protocols:** Avro, Thrift, Protobuf	**16) Application and Analytics:** Mahout , MLlib , MLbase, DataFu, R, pbdR, Bioconductor, ImageJ, Scalapack, PetSc, Azure Machine Learning, Google Prediction API, Google Translation API, mlpy, scikit-learn, PyBrain, CompLearn, Caffe, Torch, Theano, H2O, IBM Watson, Oracle PGX, GraphLab, GraphX, IBM System G, GraphBuilder(Intel), TinkerPop, Google Fusion Tables, CINET, NWB, Elasticsearch
2) **Distributed Coordination:** Google Chubby, Zookeeper, Giraffe, JGroups	**15B) Application Hosting Frameworks:** Google App Engine, AppScale, Red Hat OpenShift, Heroku, Aerobatic, AWS Elastic Beanstalk, Azure, Cloud Foundry, Pivotal, IBM BlueMix, Ninefold, Jelastic, Stackato, appfog, CloudBees, Engine Yard, CloudControl, dotCloud, Dokku, OSGi, HUBzero, OODT, Agave, Atmosphere **15A) High level Programming:** Kite, Hive, HCatalog, Tajo, Shark, Phoenix, Impala, MRQL, SAP HANA, HadoopDB, PolyBase, Presto, Google Dremel, Google BigQuery, Amazon Redshift, Drill, Pig, Sawzall, Google Cloud DataFlow, Summingbird
	14B) Streams: Storm, S4, Samza, Granules, Google MillWheel, Amazon Kinesis, LinkedIn Databus, Facebook Puma/Ptail/Scribe/ODS, Azure Stream Analytics **14A) Basic Programming model and runtime**, **SPMD, MapReduce:** Hadoop, Spark, Twister, Stratosphere (Apache Flink), Reef, Hama, Giraph, Pregel, Pegasus, Ligra, GraphChi
3) **Security & Privacy:** InCommon, Eduroam OpenStack Keystone, LDAP, Sentry, Sqrrl, OpenID, SAML OAuth	**13) Inter process communication Collectives, point-to-point, publish-subscribe:** MPI, Harp, Netty, ZeroMQ, ActiveMQ, RabbitMQ, NaradaBrokering, QPid, Kafka, Kestrel, JMS, AMQP, Stomp, MQTT, **Public Cloud:** Amazon SNS, Lambda, Google Pub Sub, Azure Queues, Event Hubs
	12) In-memory databases/caches: Gora (general object from NoSQL), Memcached, Redis (key value), Hazelcast, Ehcache, Infinispan
	12) Object-relational mapping: Hibernate, OpenJPA, EclipseLink, DataNucleus, ODBC/JDBC
	12) Extraction Tools: UIMA, Tika
4) **Monitoring:** Ambari, Ganglia, Nagios, Inca	**11C) SQL(NewSQL):** Oracle, DB2, SQL Server, SQLite, MySQL, PostgreSQL, Galera Cluster, SciDB, Rasdaman, Apache Derby, Google Cloud SQL, Azure SQL, Amazon RDS, Google F1, IBM dashDB, N1QL, BlinkDB
	11B) NoSQL: Lucene, Solr, Solandra, Voldemort, Riak, Berkeley DB, MongoDB, Espresso, CouchDB, Couchbase, IBM Cloudant, HBase, Google Bigtable, Megastore and Spanner, Accumulo, Cassandra, RYA, Sqrrl, Neo4J, Yarcdata, AllegroGraph, Facebook Tao, Titan:db, Jena, Sesame **Public Cloud:** Azure Table, Amazon Dynamo, Google DataStore
21 layers Over 300 Software Packages **9 March 2015**	**11A) File management:** iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet
	10) Data Transport: BitTorrent, HTTP, FTP, SSH, Globus Online (GridFTP), Flume, Sqoop
	9) Cluster Resource Management: Mesos, Yarn, Helix, Llama, Google Omega, Facebook Corona, Celery, HTCondor, SGE, OpenPBS, Moab, Slurm, Torque, Globus Tools, Pilot Jobs
	8) File systems: HDFS, Swift, Haystack, f4, Cinder, Ceph, FUSE, Gluster, Lustre, GPFS, GFFS **Public Cloud:** Amazon S3, Azure Blob, Google Cloud Storage
	7) Interoperability: Libvirt, Libcloud, JClouds, TOSCA, OCCI, CDMI, Whirr, Saga, Genesis
	6) DevOps: Docker, Puppet, Chef, Ansible, SaltStack, Boto, Cobbler, Xcat, Razor, CloudMesh, Juju, Foreman, OpenStack Heat, Rocks, Cisco Intelligent Automation for Cloud, Ubuntu MaaS, Facebook Tupperware, AWS OpsWorks, OpenStack Ironic, Google Kubernetes, Buildstep, Gitreceive
	5) IaaS Management from HPC to hypervisors: Xen, KVM, Hyper-V, VirtualBox, OpenVZ, LXC, Linux-Vserver, OpenStack, OpenNebula, Eucalyptus, Nimbus, CloudStack, CoreOS, VMware ESXi, vSphere and vCloud, Amazon, Azure, Google and other public Clouds, **Networking:** Google Cloud DNS, Amazon Route 53

Figure 2 collects together much existing relevant data processing software coming from either HPC or commodity sources. This is termed HPC-ABDS as many critical core components of the commodity stack (such as Hadoop and Hbase) come from open source projects while HPC is needed to bring performance and other parallel computing capabilities [11]. Note that Apache is the largest but not only source of open source software; we believe Apache Foundation is a critical leader in the Big Data open source software movement and use it to label for full big data software ecosystem. The figure also includes proprietary systems as they illustrate key capabilities and often motivate open source equivalents.

The software is broken up into layers so that one can discuss software systems in smaller groups. The layers where there is especial opportunity to integrate HPC are colored green in figure. We note that data systems that we construct from this software can run interoperably on virtualized or non-virtualized environments aimed at key scientific data analysis problems. Most of ABDS emphasizes scalability but not performance and one of our goals is to produce high performance environments. Here there is clear need for better node performance and support of accelerators like Xeon-Phi and GPU's. Figure 1 contrasts modern ABDS and HPC stacks illustrating most of the 21 layers and labelling on left with layer number used in Figure 1. The omitted layers in Figure 1 are Interoperability, DevOps, Monitoring and Security (layers 7, 6, 4, 3) which are all important and clearly applicable to both HPC and ABDS. We also add an extra layer "language" not discussed in Figure 2.

Lower layers where HPC can make a major impact include scheduling where Apache technologies like Yarn and Mesos need to be integrated with the sophisticated HPC approaches. Storage is another important area where HPC distributed and parallel storage environments need to be reconciled with the "data parallel" storage seen in HDFS in many ABDS systems. However the most important issues are probably at the higher layers with data management, communication, (high layer or basic) programming, analytics and orchestration. These are areas where there is rapid commodity/commercial innovation and we briefly discuss them in order below. Much science data analysis is centered on files but we expect movement to the common commodity approaches of Object stores, SQL and NoSQL where latter has a proliferation of systems with different characteristics – especially in the data abstraction that varies over row/column, key-value, graph and documents. Note recent developments at the programming layer like Apache Hive and Drill, which offer high-layer access models like SQL implemented on scalable NoSQL data systems. The communication layer includes Publish-subscribe technology used in many approaches to streaming data as well the HPC communication technologies (MPI) which are much faster than most default Apache approaches but can be added [12] to some systems like Hadoop whose modern version is modular and allows plug-ins for HPC stalwarts like MPI and sophisticated load balancers. The programming layer includes both the classic batch processing typified by Hadoop and streaming by Storm. The programming offerings differ in approaches to data model (key-value, array, pixel, bags, graph), fault tolerance and communication. The trade-offs here have major performance issues. You also see systems like Apache Pig offering data parallel interfaces. At the high layer we see both programming models and Platform as a Service toolkits where the Google App Engine is well known but there are many entries including the recent BlueMix from IBM. The orchestration or workflow layer has seen an explosion of activity in the commodity space although with systems like Pegasus, Taverna, Kepler, Swift and IPython, HPC has substantial experience. There are commodity orchestration dataflow systems like Tez and projects like Apache Crunch with a data parallel emphasis based on ideas from Google FlumeJava. A modern version of the latter presumably underlies Google's recently announced Cloud Dataflow that unifies support of multiple batch and streaming components; a capability that we expect to become common. The implementation of the analytics layer depends on details of orchestration and especially programming layers but probably most important are quality parallel algorithms. As many machine learning algorithms involve linear algebra, HPC expertise is directly applicable as is fundamental mathematics needed to develop O(NlogN) algorithms for analytics that are naively O(N^2). Streaming algorithms are an important new area of research

2. The 21 Layers in Detail

Layer 1) Message Protocols: *Avro, Thrift, Protobuf*

This layer is unlikely to directly visible in many applications as used in "underlying system". Thrift and Protobuf have similar functionality and are used to build messaging protocols with data syntax dependent interfaces between components (services) of system. Avro always carries schema with messages so that they can be processed generically without syntax specific interfaces.

Layer 2) Distributed Coordination: *Google Chubby, Zookeeper, Giraffe, JGroups*

Zookeeper is likely to be used in many applications as it is way that one achieves consistency in distributed systems – especially in overall control logic and metadata. It is for example used in Apache Storm to coordinate distributed streaming data input with multiple servers ingesting data from multiple sensors. Zookeeper is based on the original Google Chubby and there are several projects extending Zookeeper such as the Giraffe system. JGroups is less commonly used and is very different; it builds secure multi-cast messaging with a variety of transport mechanisms.

Layer 3) Security & Privacy: *InCommon, OpenStack Keystone, LDAP, Sentry, Sqrrl*

Security & Privacy is of course a huge area present implicitly or explicitly in all applications. It covers authentication and authorization of users and the security of running systems. In the Internet there are many authentication systems with sites often allowing you to use Facebook, Microsoft , Google etc. credentials. InCommon, operated by Internet2, federates research and higher education institutions, in the United States with identity management and related services while Eduroam has international scope. Such federated identity based authentication and authorization is made possible because of open standards like OpenID, SAML and OAuth.

LDAP is a simple database (key-value) forming a set of distributed directories recording properties of users and resources according to X.500 standard. It allows secure management of systems. OpenStack Keystone is a role-based authorization and authentication environment to be used in OpenStack private clouds. Sqrrl comes from a startup company spun off the US National Security Agency. It focusses on analyzing big data using Accumulo (layer 11B) to identify security issues such as cybersecurity incidents or suspicious data linkages.

Layer 4) Monitoring: *Ambari, Ganglia, Nagios, Inca*

Here Apache Ambari is aimed at installing and monitoring Hadoop systems and there are related tools at layers 6 and 15B. Very popular are the similar Nagios and Ganglia, which are system monitors with ability to gather metrics and produce alerts for a wide range of applications. Inca is a higher layer system allowing user reporting of performance of any sub system. Essentially all deployed systems use monitoring but most users do not add custom reporting.

Layer 5) IaaS Management from HPC to hypervisors: *Xen, KVM, Hyper-V, VirtualBox, OpenVZ, LXC, Linux-Vserver, OpenStack, OpenNebula, Eucalyptus, Nimbus, CloudStack, CoreOS, VMware ESXi, vSphere and vCloud, Amazon, Azure, Google and other public (general access) Clouds, Google Cloud DNS, Amazon Route 53*

Technologies at layer 5 underlie all applications although they may not be apparent to users. Layer 5 includes 4 major hypervisors Xen, KVM, Hyper-V and VirtualBox and the alternative virtualization approach through Linux containers OpenVZ, LXC and Linux-Vserver. OpenStack, OpenNebula, Eucalyptus, Nimbus and Apache CloudStack are leading virtual machine managers with OpenStack most popular in US and OpenNebula in Europe (for researchers). These systems manage many aspects of virtual machines including computing, security, clustering, storage and networking; in particular OpenStack has an impressive number of subprojects (16 in March 2015). As a special case there is “bare-metal” i.e. the null hypervisor, which is discussed at layer 6. The DevOps (layer 6) technology Docker is playing an increasing role as a Linux container. CoreOS is a recent lightweight version of Linux customized for containers and Docker in particular. The public clouds Amazon, Azure and Google have their own solution and it is possible to move machine images between these different environments.

Layer 6) DevOps: *Docker, Puppet, Chef, Ansible, SaltStack, Boto, Cobbler, Xcat, Razor, CloudMesh, Juju, Foreman, OpenStack Heat, Rocks, Cisco Intelligent Automation for Cloud, Ubuntu MaaS, Facebook Tupperware, AWS OpsWorks, OpenStack Ironic, Google Kubernetes, Buildstep, Gitreceive*

This layer describes technologies and approaches that automate the deployment, installation and life-cycle of software systems and underlies “software-defined systems”. At Indiana University, we integrate tools together in Cloudmesh – Libcloud, Cobbler (becoming OpenStack Ironic), Chef, Docker, Slurm, Ansible, Puppet and Celery. We saw the container support system Docker earlier in layer 5.

Puppet, Chef, Ansible and SaltStack are leading configuration managers allowing software and their features to be specified. Juju and OpenStack Heat extend this to systems or virtual clusters of multiple different software components. Cobbler, Xcat, Razor, Ubuntu MaaS, and OpenStack Ironic address bare-metal provisioning and enable IaaS with hypervisors, containers or bare-metal. Foreman is a popular general deployment environment while Boto provides from a Python interface, DevOps for all (around 40 March 2015) the different AWS public cloud Platform features. Rocks is a well-regarded cluster deployment system aimed at HPC with software configurations specified in rolls, Cisco Intelligent Automation for Cloud is a commercial offering from Cisco that directly includes networking issues. Tupperware is used by Facebook to deploy containers for their production systems while AWS OpsWorks is new Amazon capability for automating use of AWS. Google Kubernetes focusses on cluster management for Docker while Buildstep and Gitreceive are part of the Dokku application hosting framework (layer 15B) for Docker.

Layer 7) Interoperability: *Libvirt, Libcloud, JClouds, TOSCA, OCCI, CDMI, Whirr, SAGA, Genesis II*

This layer has both standards and interoperability libraries for services, compute, virtualization and storage. Libvirt provides common interfaces at the hypervisor level while Libcloud and JClouds provide this at the higher cloud provider level. TOSCA is an interesting DevOps standard specifying the type of system managed by OpenStack Heat. OCCI and CDMI provide cloud computing and storage interfaces respectively while Whirr provides cloud-neutral interfaces to services. SAGA and Genesis II come from HPC community and provide standards and their implementations for distributed computing and storage.

Layer 8) File systems: *HDFS, Swift, Amazon S3, Azure Blob, Google Cloud Storage, Haystack, f4, Cinder, Ceph, FUSE, Gluster, Lustre, GPFS, GFFS*

One will use files in most applications but the details may not be visible to the user. Maybe you interact with data at layer of a data management system like iRODS or an Object store (OpenStack Swift or Amazon S3). Most science applications are organized around files; commercial systems at a higher layer. For example originally Facebook directly used a basic distributed file systems (NFS) to store images but in Haystack replaced this with a customized object store which was refined in f4 to accommodate variations in image usage with “cold”, “warm” and “hot” data. HDFS realizes the goal of bring computing to data and has a distributed data store on the same nodes that perform computing; it underlies the Apache Hadoop ecosystem. Openstack Cinder implements the Block store used in Amazon Elastic Block Storage EBS, Azure Files and Google Persistent Storage. This is analogous to disks accessed directly on a traditional non-cloud environment whereas Swift, Amazon S3, Azure Blob and Google Cloud Storage implement backend object stores. Lustre is a major HPC shared cluster file system with Gluster as an alternative. FUSE is a user level file system used by Gluster. Ceph is a distributed file system that projects object, block, and file storage paradigms to the user. GPFS or Global Parallel File System is a IBM parallel file system optimized to support MPI-IO and other high intensity parallel I/O scenarios. GFFS or Global Federated File System comes from Genesis II (layer 7) and provides a uniform view of a set of distributed file systems.

Layer 9) Cluster Resource Management: *Mesos, Yarn, Helix, Llama, Google Omega, Facebook Corona, Celery, HTCondor, SGE, OpenPBS, Moab, Slurm, Torque, Globus Tools, Pilot Jobs*

You will certainly need cluster management in your application although often this is provided by the system and not explicit to the user. Yarn from Hadoop is very popular while Mesos from UC Berkeley is similar to Yarn and is also well used. Apache Helix originating in LinkedIn is similar to Mesos and Yarn. Llama from Cloudera runs above Yarn and achieves lower latency by switching use of long lived processes. Google and Facebook certainly face job management at a staggering scale and Omega and Corona respectively are proprietary systems along the lines of Yarn and Mesos. Celery is built on RabbitMQ and supports the “master-worker” model in a similar fashion to Azure worker role with Azure queues.

Slurm is a basic HPC system as are Moab, Torque, SGE, OpenPBS while Condor also well known for scheduling of Grid applications. Many systems are in fact collections of clusters as in data centers or grids. These require management and scheduling across many clusters; the latter is termed metascheduling. These are addressed by the Globus Toolkit and Pilot jobs which originated in the Grid computing community.

Layer 10) Data Transport: *BitTorrent, HTTP, FTP, SSH, Globus Online (GridFTP), Flume, Sqoop*

BitTorrent was famous 10 years ago (2004) for accounting one third of all Internet traffic; this is dropped a factor of 10 in 2014 but still an important Peer to Peer (file sharing) protocol. Simple HTTP protocols are typically used for small data transfers while the largest one might even use the "Fedex/UPS" solution of transporting disks between sites. SSH and FTP are old well established Internet protocols underlying simple data transfer. Apache Flume is aimed at transporting log data and Apache Sqoop at interfacing distributed data to Hadoop.

Globus Online or GridFTP is dominant and successful system for the HPC community.

Data transport is often not highlighted as it runs under the covers but is often quoted as a major bottleneck.

Layer 11A) File management: *iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet*

The data management layer 11 is a critical area for nearly all applications as it captures areas of file, object, NoSQL and SQL data management. The many entries in area testify to variety of problems (graphs, tables, documents, objects) and importance of efficient solution. Just a little while ago, this area was dominated by SQL databases and file managers. We divide this layer into 3 subparts; management and data structures for file in 11A; the cloud NoSQL systems in 11B and the traditional SQL systems in layer 11C, which also includes the recent crop of NewSQL systems that overlap with layer 15A.

It is remarkable that the Apache stack does not address file management (as object stores are used instead of file systems) and the HPC system iRODS is major system to manage files and their metadata. This layer also includes important file (data) formats. NetCDF and CDF are old but still well used formats supporting array data, as is HDF or Hierarchical Data Format. The earth science domain uses OPeNDAP and the astronomers FITS.

RCFile (Row Column File) and ORC are new formats introduced with Hive (layer 15A) while Parquet based on Google Dremel is used for column storage in many major systems in layers 14A and 15A.

Layer 11B) NoSQL: *Lucene, Solr, Solandra, Voldemort, Riak, Berkeley DB, Azure Table, Amazon Dynamo, Google DataStore, MongoDB, Espresso, CouchDB, Couchbase, IBM Cloudant, HBase, Google Bigtable, Megastore and Spanner, Accumulo, Cassandra, RYA, Sqrrl, Neo4J, Yarcdata, AllegroGraph, Facebook Tao, Titan:db, Jena, Sesame*

NoSQL systems can be divided into six important styles: Tools, Key-value stores, Document-based stores, Column-based store, Graph-based stores and Triple stores.

Tools include Apache Lucene providing information-retrieval; Apache Solr uses Lucene to build a fast search engine while Apache Solandra adds Cassandra as a backend to Solr.

Key-value stores have a hash table of keys and values and include Voldemort from LinkeIn; Riak uses Solr for search and is based on Amazon Dynamo; Berkeley DB comes from Oracle; Azure Table, Amazon Dynamo, and Google DataStore are the dominant public cloud NoSQL key-value stores.

Document-based stores manage documents made up of tagged elements and include MongoDB which is best known system in this class. Espresso comes from LinkedIn and uses Helix (layer 9); Apache CouchDB has a variant Couchbase that adds caching (memcached) features; IBM Cloudant is a key part of IBM's cloud offering.

Column-based stores have data elements that just contain data from one column as pioneered by Google Bigtable which inspires Apache Hbase; Google Megastore and Spanner build on Bigtable to provide capabilities that interpolate between NoSQL and SQL and can get scalability of NoSQL and ease of use of SQL; Apache Cassandra comes from Facebook; Apache Accumulo is also popular and RYA builds a triple store on top of it; Sqrrl is built on top of Accumulo to provide graph capabilities and security applications.

Graph-based Stores: Neo4j is most popular graph database; Yarcdata Urika is supported by Cray shared memory machines and allows SPARQL queries as does AllegroGraph, which is written in Lisp and is integrated with Solr and MongoDB; Facebook TAO (The Associations and Objects) supports their specific problems with massive scaling; Titan:db is an interesting graph database and integrates with

Cassandra, HBase, BerkeleyDB, TinkerPop graph stack (layer 16), Hadoop (layer 14A), ElasticSearch (layer 16), Solr and Lucene (layer 11B)

Triple stores: The last category is a special case of the graph database specialized to the triples (typically resource, attribute and attribute value) that one gets in the RDF approach. Apache Jena and Sesame support storage and queries including those in SPARQL.

Layer 11C) SQL/NewSQL: *Oracle, DB2, SQL Server, SQLite, MySQL, PostgreSQL, Galera Cluster, SciDB, Rasdaman, Apache Derby, Google Cloud SQL, Azure SQL, Amazon RDS, Google F1, IBM dashDB, N1QL, BlinkDB*

Layer 11C only lists a few of the traditional relational databases but includes the NewSQL area, which is also seen in systems at layer 15A. NewSQL combines a rigorous SQL interface with the scalability of MapReduce style infrastructure. Oracle, IBM DB2 and Microsoft SQL Server are of course major commercial databases but the amazing early discovery of cloud computing was that their architecture, optimized for transaction processing, was not the best for many cloud applications. Traditional databases are still used with Apache Derby, SQLite, MySQL, and PostgreSQL being important low-end open source systems. Galera Cluster is one of several examples of a replicated parallel database built on MySQL. SciDB and Rasdaman stress another idea; good support for the array data structures we introduced in layer 11A. Google Cloud SQL, Azure SQL, Amazon RDS, are the public cloud traditional SQL engines with Azure SQL building on Microsoft SQL server. N1QL illustrates an important trend and is designed to add SQL queries to the NoSQL system Couchbase. Google F1 illustrates the NewSQL concept building a quality SQL system on the Spanner system described in layer 11B. IBM dashDB similarly offers warehouse capabilities built on top of the NoSQL Cloudant which is a derivative of CouchDB (again layer 11B). BlinkDB is a research database exploring sampling to speed up queries on large datasets.

Layer 12) In-memory databases&caches: *Gora, Memcached, Redis, Hazelcast, Ehcache, Infinispan* **/ Object-relational mapping:** *Hibernate, OpenJPA, EclipseLink, DataNucleus, ODBC/JDBC* **/ Extraction Tools:** *UIMA, Tika*

This layer represents another important area addressing several important capabilities. Firstly Memcached (best known and used by GAE), Redis (an in-memory key value store), Hazelcast, Ehcache, Infinispan enable caching to put as much processing as possible in memory. This is an important optimization with Gartner highlighting in several recent hype charts with In-Memory database management systems and Analytics. UIMA and Tika are conversion tools with former well known from its use by Jeopardy winning IBM Watson system. Gora supports generation of general object data structures from NoSQL. Hibernate, OpenJPA, EclipseLink and DataNucleus are tools for persisting Java in-memory data to relational databases.

Layer 13) Inter process communication Collectives, point-to-point, publish-subscribe, MPI: *MPI, Harp, Netty, ZeroMQ, ActiveMQ, RabbitMQ, NaradaBrokering, QPid, Kafka, Kestrel, JMS, AMQP, Stomp, MQTT, Amazon SNS and Lambda, Google Pub Sub, Azure Queues and Event Hubs,*

This layer describes the different communication models used by the systems in layers 14 and 15) below. One has communication between the processes in parallel computing and communication between data sources and filters. There are important trade-offs between performance, fault tolerance, and flexibility. There are also differences that depend on application structure and hardware. MPI from the HPC domain, has very high performance and has been optimized for different network structures while its use is well understood across a broad range of parallel algorithms. Data is streamed between nodes (ports) with latencies that can be as low as a microsecond. This contrasts with disk access with latencies of 10 milliseconds and event brokers of around a millisecond corresponding to the significant software supporting messaging systems. Hadoop uses disks to store data in between map and reduce stages, which removes synchronization overheads and gives excellent fault tolerance at cost of highest latency. Harp brings MPI performance to Hadoop with a plugin.

There are several excellent publish-subscribe messaging systems that support publishers posting messages to named topics and subscribers requesting notification of arrival of messages at topics. The systems differ in message protocols, API, richness of topic naming and fault tolerance including message delivery guarantees. Apache has Kafka from LinkedIn with strong fault tolerance, ActiveMQ and QPid. RabbitMQ and NaradaBrokering have similar good performance to ActiveMQ. Kestrel from Twitter is simple and fast while Netty is built around Java NIO and can support protocols like UDP which are useful for messaging with media streams as well as HTTP. ZeroMQ provides fast inter-process communication with software multicast. Message queuing is well supported in commercial clouds but with different software environments; Amazon Simple Notification Service, Google Pub-Sub, Azure queues or service-bus queues. Amazon offers a new event based computing model Lambda while Azure has Event Hubs built on the service bus to support Azure streaming analytics in layer 14B.

There are important messaging standards supported by many of these systems. JMS or Java Message Service is a software API that does not specify message nature. AMQP (Advanced Message Queuing Protocol) is best known message protocol while STOMP (Simple Text Oriented Messaging Protocol) is a particularly simple and HTTP in style without supporting topics. MQTT (Message Queue Telemetry Transport) comes from the Internet of Things domain and could grow in importance for machine to machine communication.

Layer 14A) Basic Programming model and runtime, SPMD, MapReduce: *Hadoop, Spark, Twister, Stratosphere (Apache Flink), Reef, Hama, Giraph, Pregel, Pegasus, Ligra, GraphChi*

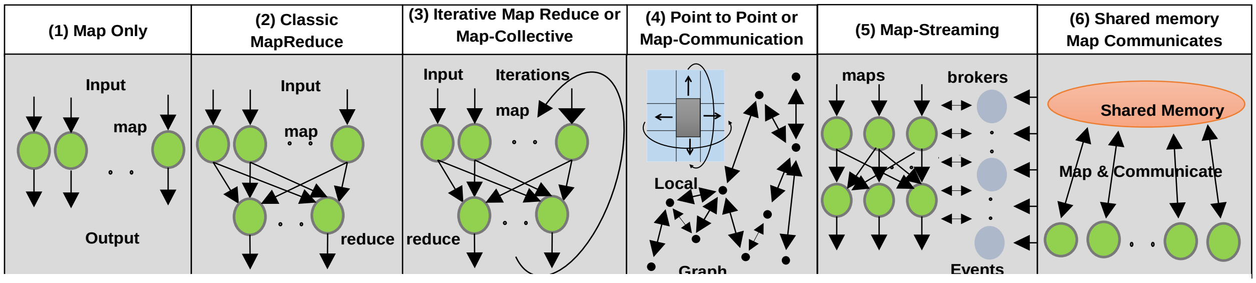

Figure 3: 6 distinctive programming models labelled with map-X syntax and supported at layer 14

Most applications use capabilities at layers 14 which we divide into the classic or batch programming in 14A and the streaming area 14B that has grown in attention recently. This layer implements the programming models shown in Fig. 3. Layer 14B supports the Map-Streaming model which is category 5 of Fig. 3. Layer 14A focusses on the first 4 categories of this figure while category 6 is included because shared memory is important in some graph algorithms. Hadoop in some sense created this layer although its programming model had been known for a long time but not articulated as brilliantly as was done by Google for MapReduce. Hadoop covers categories 1 and 2 of Fig. 3 and with the Harp plug-in categories 3 and 4. Other entries here have substantial overlap with Spark and Twister (no longer developed) being pioneers for Category 3 and Pregel with an open source version Giraph supporting category 4. Pegasus also supports graph computations in category 4. Hama is an early Apache project with capabilities similar to MPI with Apache Flink and Reef newcomers supporting all of categories 1-4. Flink supports multiple data API's including graphs with its Spargel subsystem. Reef is optimized to support machine learning. Ligra and GraphChi are shared memory graph processing frameworks (category 6 of Fig. 3) with GraphChi supporting disk-based storage of graph.

Layer 14B) Streaming: *Storm, S4, Samza, Granules, Google MillWheel, Amazon Kinesis, LinkedIn Databus, Facebook Puma/Ptail/Scribe/ODS, Azure Stream Analytics*

Figure 3, category 5, sketches the programming model at this layer while figure 4 gives it more detail for Storm, which being open source and popular, is the best archetype for this layer. There is some mechanism to gather and buffer data, which for Apache Storm is a publish-subscribe environment such as Kafka, RabbitMQ or ActiveMQ. Then there is a processing phase delivered in Storm as "bolts" implemented as dataflow but which can invoke parallel processing such as Hadoop. The bolts then deliver their results to a similar buffered environment for further processing or storage. Apache has three similar environments Storm, Samza and S4 which were donated by Twitter, LinkedIn and Yahoo respectively. S4 features a built-in key-value store. Granules from Colorado State University has a similar model to Storm.

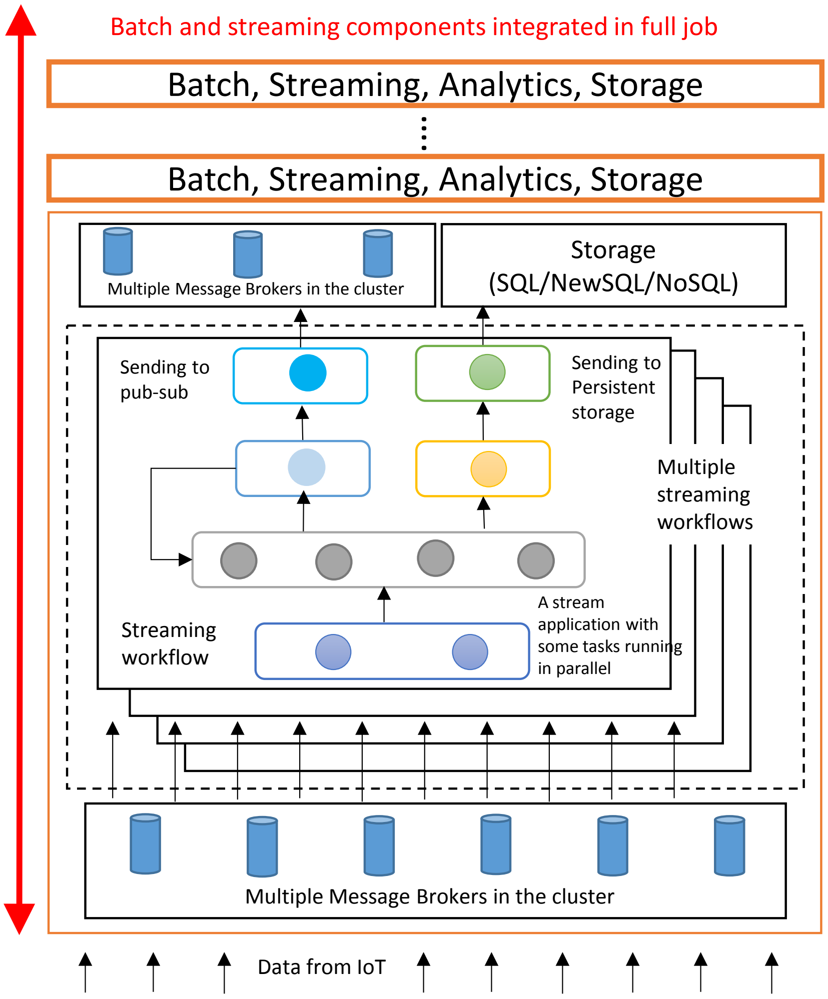
Batch and streaming components integrated in full job
Batch, Streaming, Analytics, Storage
Batch, Streaming, Analytics, Storage
Multiple Message Brokers in the cluster
Storage
(SQL/NewSQL/NoSQL)
Sending to pub-sub
Sending to Persistent storage
Multiple streaming workflows
A stream application with some tasks running in parallel
Streaming workflow
Multiple Message Brokers in the cluster
Data from IoT

Figure 4: Apache Storm processing data from Internet of Things (IoT)

The other entries are commercial systems with LinkedIn Databus and Facebook Puma/ Ptail/ Scribe/ ODS supporting internal operations of these companies for examining in near real-time the logging and response of their web sites. Kinesis, MillWheel and Azure Stream Analytics are services offered to customers of the Amazon, Google and Microsoft clouds respectively. Interestingly none of these uses the Apache functions (Storm, Samza, S4) although these run well on commercial clouds.

Layer 15A) High layer Programming: *Kite, Hive, HCatalog, Tajo, Shark, Phoenix, Impala, MRQL, SAP HANA, HadoopDB, PolyBase, Presto, Google Dremel, Google BigQuery, Amazon Redshift, Drill, Pig, Sawzall, Google Cloud DataFlow, Summingbird*

Components at this layer are not required but are very interesting and we can expect great progress to come both in improving them and using them. There are several "SQL on MapReduce" software systems with Apache Hive (originally from Facebook) as best known. Hive illustrates key idea that MapReduce supports parallel database actions and so SQL systems built on Hadoop can outperform traditional databases due to scalable parallelism. Presto is another Facebook system supporting interactive queries rather than Hive's batch model. Apache HCatalog is a table and storage management layer for Hive. Other SQL on MapReduce systems include Shark (using Spark not Hadoop), MRQL (using Hama, Spark or Hadoop for complex analytics as queries), Tajo (warehouse) Impala (Cloudera), HANA (real-time SQL from SAP), HadoopDB (true SQL on Hadoop), and PolyBase (Microsoft SQL server on Hadoop). A different approach is Kite which is a toolkit to build applications on Hadoop. Note layer 11C) lists "production SQL data bases" including some NewSQL systems with architectures similar to software described here.

Google Dremel supported SQL like queries on top of a general data store including unstructured and NoSQL systems. This capability is now offered by Apache Drill, Google BigQuery and Amazon Redshift. Apache Phoenix exposes HBase functionality through a SQL interface that is highly optimized. Pig and Sawzall offer data parallel programming models on top of Hadoop (Pig) or Google MapReduce (Sawzall). Pig shows particular promise as fully open source with DataFu providing analytics libraries on top of Pig. Summingbird builds on Hadoop and supports both batch (Cascading) and streaming (Storm) applications. Google Cloud Dataflow provides similar integrated batch and streaming interfaces.

Layer 15B) Application Hosting Frameworks: *Google App Engine, AppScale, Red Hat OpenShift, Heroku, Aerobatic, AWS Elastic Beanstalk, Azure, Cloud Foundry, Pivotal, IBM BlueMix, Ninefold,* *Jelastic, Stackato, appfog, CloudBees, Engine Yard, CloudControl, dotCloud, Dokku, OSGi, HUBzero, OODT, Agave, Atmosphere*

This layer is exemplified by Google App Engine GAE and Azure where frameworks are called "Platform as a Service" PaaS but now there are many "cloud integration/development environments". The GAE style framework offers excellent cloud hosting for a few key languages (often PHP, JavaScript Node.js, Python, Java and related languages) and technologies (covering SQL, NoSQL, Web serving, memcached, queues) with hosting involving elasticity, monitoring and management. Related capabilities are seen in AWS Elastic Beanstalk, AppScale, appfog, OpenShift (Red Hat), Heroku (part of Salesforce) while Aerobatic is specialized to single web page applications. Pivotal offers a base open source framework Cloud Foundry that is used by IBM in their BlueMix PaaS but is also offered separately as Pivotal Cloud Foundry and Web Services. The major vendors AWS, Microsoft, Google and IBM mix support of general open source software like Hadoop with proprietary technologies like BigTable (Google), Azure Machine Learning, Dynamo (AWS) and Cloudant (IBM). Jelastic is a Java PaaS that can be hosted on the major IaaS offerings including Docker. Similarly Stackato from ActiveState can be run on Docker and Cloud Foundry. CloudBees recently switched from general PaaS to focus on Jenkins continuous integration services. Engine Yard and Cloud Control focus on managing (scaling, monitoring) the entire application. The latter recently purchased the dotCloud PaaS from Docker, which company was originally called dotCloud but changed name and focus due to huge success of their "support" product Docker. Dokku is a bash script providing Docker Heroku compatibility based on tools Gitreceive and Buildstep. This layer also includes toolsets that are used to build web environments OSGi, HUBzero and OODT. Agave comes from the iPlant Collaborative and offers "Science as a Service" building on iPlant Atmosphere cloud services with access to 600 plant biology packages. Software at this layer has overlaps with layer 16, application libraries, and layer 17, workflow and science gateways.

Layer 16) Application and Analytics: *Mahout , MLlib , MLbase, DataFu, R, pbdR, Bioconductor, ImageJ, Scalapack, PetSc, Azure Machine Learning, Google Prediction API, Google Translation API, mlpy, scikit-learn, PyBrain, CompLearn, Caffe, Torch, Theano, H2O, IBM Watson, Oracle PGX, GraphLab, GraphX, IBM System G, GraphBuilder(Intel), TinkerPop, Google Fusion Tables, CINET, NWB, Elasticsearch*

This is the "business logic" of application and where you find machine learning algorithms like clustering, recommender engines and deep learning. Mahout, MLlib, MLbase are in Apache for Hadoop and Spark processing while the less well-known DataFu provides machine learning libraries on top of Apache Pig. R with a custom scripting language, is a key library from statistics community with many domain specific libraries such as Bioconductor, which has 936 entries in version 3.0. Image processing (ImageJ) in Java and High Performance Computing HPC (Scalapack and PetSc) in C++/Fortran also have rich libraries. pbdR uses Scalapack to add high performance parallelism to R which is well known for not achieving high performance often necessary for Big Data. Note R without modification will address the important pleasingly parallel sector with scaling number of independent R computations. Azure Machine Learning, Google Prediction API, and Google Translation API represent machine learning offered as a service in the cloud.

The array syntaxes supported in Python and Matlab make them like R attractive for analytics libraries. mlpy, scikit-learn, PyBrain, and CompLearn are Python machine-learning libraries while Caffe (C++, Python, Matlab), Torch (custom scripting), DL4J (Java) and Theano (Python) support the deep learning area with has growing importance and comes with GPU's as key compute engine. H2O is a framework using R and Java that supports a few important machine learning algorithms including deep learning, K-means, and Random Forest and runs on Hadoop and HDFS. IBM Watson applies advanced natural language processing, information retrieval, knowledge representation, automated reasoning, and machine learning technologies to answer questions. It is being customized in hundreds of areas following its success in the game Jeopardy

There are several important graph libraries including Oracle PGX, GraphLab (CMU), GraphBuilder (Intel), GraphX (Spark based), and TinkerPop (open source group). IBM System G has both graph and Network Science libraries and there are also libraries associated with graph frameworks (Giraph and Pegasus) at Layer 14A. CINET and Network Workbench focus on Network science including both graph construction and analytics.

Google Fusion Tables focus on analytics for Tables including map displays. ElasticSearch combines search (second only to Solr in popularity) with an analytics engine Kibana.

You will nearly always need software at this layer

Layer 17) Workflow-Orchestration: *ODE, ActiveBPEL, Airavata, Pegasus, Kepler, Swift, Taverna, Triana, Trident, BioKepler, Galaxy, IPython, Dryad, Naiad, Oozie, Tez, Google FlumeJava, Crunch, Cascading, Scalding, e-Science Central, Azure Data Factory, Google Cloud Dataflow, NiFi (NSA)*

This layer implements orchestration and integration of the different parts of a job. This integration is typically specified by a directed data-flow graph and a simple but important is a pipeline of the different stages of a job. Essentially all problems involve the linkage of multiple parts and the terms orchestration or workflow are used to describe the linkage of these different parts. The “parts” (components) involved are “large” and very different from the much smaller parts involved in parallel computing involving MPI or Hadoop. On general principles, communication costs decrease in comparison to computing costs as problem sizes increase. So orchestration systems are not subject to the intense performance issues we saw in layer 14. Often orchestration involves linking of distributed components. We can contrast PaaS stacks which describe the different services or functions in a single part with orchestration that describes the different parts making up a single application (job). The trend to “Software as a Service” clouds the distinction as it implies that a single part may be made up of multiple services. The messaging linking services in PaaS is contrasted with dataflow linking parts in orchestration. Note that often the orchestration parts often communicate via disk although faster streaming links are also common.

Orchestration in its current form originated in the Grid and service oriented communities with the early importance of OASIS standard BPEL (Business Process Execution Language) illustrated by ActiveBPEL which was last updated in 2008. BPEL did not emphasize dataflow and was not popular in Grid community. Pegasus, Kepler, and Taverna are perhaps the best known Grid workflow systems with recently Galaxy and BioKepler popular in bioinformatics. The workflow system interface is either visual (link programs as bubbles with data flow) or as an XML or program script. The latter is exemplified by the Swift customized scripting system and the growing use of Python. Apache orchestration of this style is seen in ODE and Airavata with latter coming from Grid research at Indiana University. Recently there has been a new generation of orchestration approaches coming from Cloud computing and covering a variety of approaches. Dryad, Naiad and the recent NiFi support a rich dataflow model which underlies orchestration. These systems tend not to address the strict synchronization needed for parallel computing and that limits their breadth of use. Apache Oozie and Tez link well with Hadoop and are alternatives to high layer programming models like Apache Pig. e-Science Central from Newcastle, the Azure Data Factory and Google Cloud Dataflow focus on the end to end user solution and combine orchestration with well-chosen PaaS features. Google FlumeJava and its open source relative Apache Crunch are sophisticated efficient Java orchestration engines. Cascading, PyCascading and Scalding offer Java, Python and Scala toolkits to support orchestration. One can hope to see comparisons and integrations between these many different systems.

3. Conclusions

We have found it very fruitful to use HPC-ABDS, which merges High Performance Computing with the Commodity Big Data Stack ABDS. We have applied it to design high performance run time [1, 3, 12], understand the mapping of applications into the software stack [2] and to delineate a systematic big data benchmark approach [8, 9]. In this paper, we have aimed at an overall discussion of all layers which can be used in many applications of HPC-ABDS and in fact in studies of just the Commodity big data stack. Our summary highlights areas such as those colored green in fig. 2 where there is particular synergy between HPC and ABDS software. For example, there have been several recent ABDS orchestration systems that do not seem to build on the related workflow systems in layer 17 from HPC. One sees relatively sparse coverage in data transport and file management layers 10 and 11C). Areas where HPC could usefully learn from ABDS include storage layers 8 and 11 (use objects not files?), streams (where ABDS is pioneering interesting programming models) and PaaS or hosting environments which are popular in clouds and use an approach that can be broadly used including in HPC.

References

1. Judy Qiu, Shantenu Jha, Andre Luckow, and Geoffrey C.Fox, *Towards HPC-ABDS: An Initial High-Performance Big Data Stack*, in *Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data*. March 18-21, 2014. San Diego Supercomputer Center, San Diego. http://grids.ucs.indiana.edu/ptliupages/publications/nist-hpc-abds.pdf.
2. Geoffrey Fox, Judy Qiu, and Shantenu Jha, *High Performance High Functionality Big Data Software Stack*, in *Big Data and Extreme-scale Computing (BDEC)*. 2014. Fukuoka, Japan. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf.
3. Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, and Geoffrey C. Fox, *A Tale of Two Data-Intensive Approaches: Applications, Architectures and Infrastructure*, in *3rd International IEEE Congress on Big Data Application and Experience Track*. June 27- July 2, 2014. Anchorage, Alaska. http://arxiv.org/abs/1403.1528.
4. *HPC-ABDS Kaleidoscope of over 300 Apache Big Data Stack and HPC Tecnologies*. [accessed 2014 April 8]; Available from: http://hpc-abds.org/kaleidoscope/.
5. Geoffrey C.Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, *Towards an Understanding of Facets and Exemplars of Big Data Applications*, in *20 Years of Beowulf: Workshop to Honor Thomas Sterling's 65th Birthday* October 14, 2014. Annapolis http://grids.ucs.indiana.edu/ptliupages/publications/OgrePaperv9.pdf.
6. Geoffrey Fox and Wo Chang, *Big Data Use Cases and Requirements*, in *1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data* March 18 - 21, 2014. San Diego Supercomputer Center, San Diego. http://grids.ucs.indiana.edu/ptliupages/publications/NISTUseCase.pdf.
7. *NIST Big Data Use Case & Requirements*. 2013 [accessed 2015 March 1]; Available from: http://bigdatawg.nist.gov/V1_output_docs.php.
8. Geoffrey C.Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, *Ogres: A Systematic Approach to Big Data Benchmarks*, in *Big Data and Extreme-scale Computing (BDEC)* January 29-30, 2015. Barcelona. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/OgreFacets.pdf.
9. Geoffrey C. FOX , Shantenu JHA, Judy QIU, Saliya EKANAYAKE, and Andre LUCKOW, *Towards a Comprehensive Set of Big Data Benchmarks*. February 15, 2015. http://grids.ucs.indiana.edu/ptliupages/publications/OgreFacetsv9.pdf.
10. Geoffrey Fox. *Data Science Curriculum: Indiana University Online Class: Big Data Open Source Software and Projects*. 2014 [accessed 2014 December 11]; Available from: http://bigdataopensourceprojects.soic.indiana.edu/.
11. Dan Reed and Jack Dongarra. *Exascale Computing and Big Data: The Next Frontier*. 2014 [accessed 2015 March 8]; Available from: http://www.netlib.org/utk/people/JackDongarra/PAPERS/Exascale-Reed-Dongarra.pdf.
12. Bingjing Zhang, Yang Ruan, and Judy Qiu, in *IEEE International Conference on Cloud Engineering (IC2E)*. March 9-12, 2015. Tempe AZ. http://grids.ucs.indiana.edu/ptliupages/publications/HarpQiuZhang.pdf.