CINET Collaboration Report

TAK LON (Stephen) WU, Thomas Wiggins, Judy Qiu

Indiana University

October 6, 2013

(Revised September 17, 2013, August 7, 2013, January 19, 2013 and August 23, 2012)

Abstract

On-demand private cloud environment has become popular due to its flexibility and scalability. However, the complicated setup and installation process to obtain this computing resource has deterred some users. Inspired by the batch job model, we introduce dynamic provisioning software, salsaDPI, to simplify these complications, allowing domain scientists to focus on their scientific problems and run their applications on clouds with dynamic and automated installation and configurations. Based on our experience using and running IaaS (Eucalyptus and Openstack) on FutureGrid, we now support portable-cloud capabilities to automate the scientific data analysis of the CINET project. Users of this framework can easily prepare their one-click configuration plan and launch their applications in different clouds. This portable API (salsaDPI) has been successfully deployed and made available to 200 graduate-level students and IT staff at NCSA Science Cloud Summer School 2012.

Table of Contents

1. Introduction

This report provides a summary of our progress on the CINET project at Indiana University in collaboration with the Virginia Polytechnic Institute and State University (hereafter written as Virginia Tech).

Batch job has been widely adopted by scientists and commercial companies to handle daily tasks that analyze terabytes of data on different computing resources. These tasks execute well in traditional static environments such as high performance clusters. However, it is not easy to run them in a dynamic environment like clouds. Clouds provide Infrastructure-as-a-Service (IaaS), enabling users to take control of the computing nodes. The on-demand elastic model utilizes computing resources, which allows applications to scale up to hundreds of nodes and hides the complicated cluster settings. Scientists may take advantage of this computing model whenever they have an urgent deployment request for computing resources. But before users submit a request to start up compute nodes, they need to choose the type of virtualization image that is prepackaged from resource providers, such as FutureGrid [1]. Users can even bundle and upload their own image with specified software stack installation. The progress of resource selection and software configuration may be repeated daily by different cloud users [2, 3]. In addition, once the VM image is up and running, users normally run their applications and obtain the output by typing commands line by line. Although users can write scripts to run these applications remotely via ssh connection, it's hard to be automatic on this batch-like job from selecting the resources, starting the compute nodes, installing software on-the-fly once nodes are running, executing user-defined applications, and releasing the computing resources. Based on these considerations, we develop an experimental on-demand dynamic provisioning software called salsaDPI which automates the job executions running on FutureGrid cloud service, i.e. Eucalyptus and OpenStack. Moreover, in order to understand the performance behaviors of running virtual instances, we also introduce a resource monitoring interface called salsaScaler which utilizes Ganglia as a performance metrics provider to monitor user-interested metrics.

This report is structured as follows. Section 2 gives an overview of the background technologies. Section 3 discusses the related works. Section 4 describes the CINET system design. Section 5 presents design and architecture of salsaDPI. Section 6 introduces salsaScaler. Section 7 gives a use case in Science Cloud Summer School 2012. Section 8 provides the system evaluations. Section 9 discusses the possible experiments and draws the conclusion with future research directions.

2. Background

Infrastructure-as-a-Service is the basic service model of cloud computing. It provides a scalable computing environment supported by a large amount of virtual machines managed through virtualization management tools such as Eucalyptus, OpenStack, Nimbus and OpenNebula. These virtual machines are running as guest operating systems on the top of hypervisor. Here, Xen and KVM are the most common open source hypervisors utilized by the private cloud providers. In order to run applications on cloud, users may have to create and upload their own images by installing special software stack on a base image. This image generation progress is normally handled by system administrators, as it requires system background knowledge and goes through several complicated steps.

Opscode Chef [4] is a provisioning and configuration management tool that deploys cloud resources and user-specified applications throughout different IaaS clouds and physical clusters automatically. It is an open source system written in Ruby, which means the configuration plan (chef recipes) can be easily built. Also, Chef is implemented as a traditional client-server model, where it invokes system and API calls through a RESTful API service or through the default command line tool knife. Starting from Chef 0.10, chef project provides contributor programming interface to write plugins, which enables more functionalities. Knife-eucalyptus is one of the plugins that enables cloud support. It provisions compute nodes directly on various clouds such as Amazon Cloud, Eucalyptus Cloud, and OpenStack Cloud. Figure 1 shows the internal communication when chef client is provisioning compute nodes on clouds. Knife-euca and knife-openstack utilize several Ruby and Chef libraries to achieve this compute node provisioning task. They are Fog, Chef::Knife:Bootstrap, Chef::Knife:EucaBase, Chef::Json_compat, Net::Ssh, and Net::Ssh::Multi. Here, Fog Library is the key component to communicating with cloud infrastructure API. Chef::Knife:Bootstrap takes chef-client installation tasks by loading default or user-defined ruby templates based on the started VM's Operation System. In addition, this tool utilizes standard ssh library Net::Ssh and Net::Ssh::Multi to execute related software installation commands.

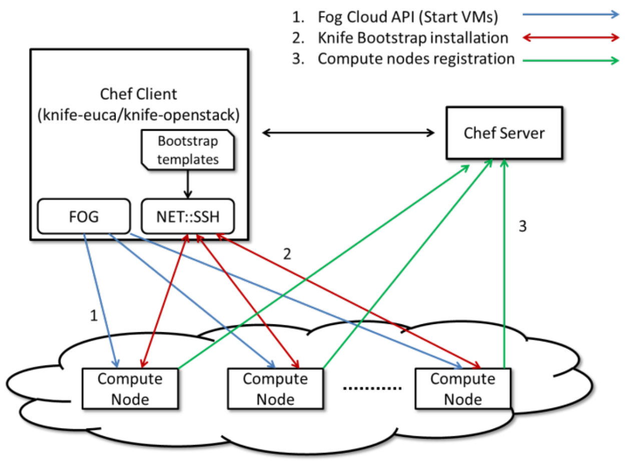

1. Fog Cloud API (Start VMs)
2. Knife Bootstrap installation
3. Compute nodes registration
Chef Client
(knife-euca/knife-openstack)
Bootstrap templates
FOG
NET::SSH
Chef Server
1
2
3
Compute Node
Compute Node
Compute Node

Figure 1. Chef knife-euca and knife-openstack internal communication

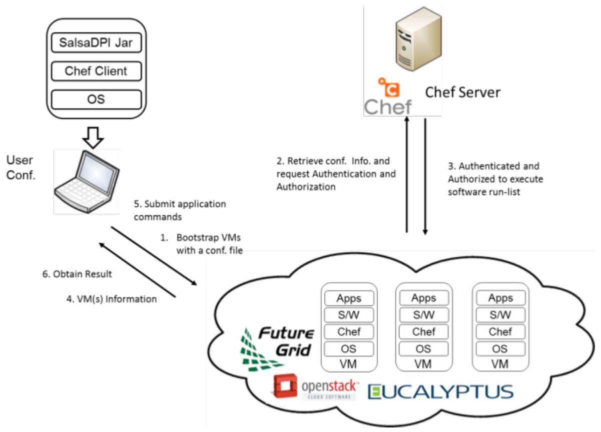

SalsaDPI Jar
Chef Client
OS
User Conf.
Chef
Chef Server
2. Retrieve conf. Info. and request Authentication and Authorization
3. Authenticated and Authorized to execute software run-list
5. Submit application commands
1. Bootstrap VMs with a conf. file
6. Obtain Result
4. VM(s) Information
Apps
S/W
Chef
OS
VM
Future Grid
openstack
EUCALYPTUS

Figure 2. salsaDPI control flow

3. Related Work

Engage [5] is a deployment management system which targets the system deployment within a distributed environment. It defines three key system components: a domain-specific component describes interdependent software dependencies; a constraint-based component constructs the software installation plan based on the interdependent software dependencies; a runtime component deploys the system and coordinates the application across multiple machines. Engage is built without integrating any existing configuration management tool such as Chef. Also, it focuses on definition of resource type, software dependencies, and software installation order in a concrete model.

Cloudinit.d [3, 6] presents a Unix-like init.d tool for launching, configuring, monitoring, and fault tolerating a set of interdependent VMs over IaaS clouds. Cloudinit.d is similar to Engage, but it provides a monitoring component to periodically check the system status before and after the first successful software deployment. This feature helps the system to detect and recover from unexpected failures in a distributed environment.

J. Klinginsmith [2] introduces a reproducible framework which was developed based on FutureGrid and Amazon Cloud. As data locality is one of the key features of Cloud Computing, his work mainly focuses on moving computational infrastructure as close as possible to the location of data in a dynamically deployed environment. The goal is to help users, especially eScience scientists, to automate their daily experiments by going through two phases: virtual machine resource allocation on-demand at infrastructure level and installation of selected software on-the-fly when resources are booting up. Also, he writes a customized Chef Plugin (with Chef 0.10+) to read different configuration parameters in order to support various cloud IaaS API. This work has been presented at NCSA Science Cloud Summer School 2012 [7] as an educational package. salsaDPI framework has been installed within a pre-packaged portable VirtualBox [8] image for this event.

4. Overview of CINET System Design

As shown in Figure 3, requests sent to FutureGrid data management and execution brokers involve jobs running on FutureGrid Eucalyptus cloud service. We provide support and work with CINET project contributor Hemanth Makkapati from Virginia Tech regarding system design and testing several potential components which may be integrated into data management and execution brokers. These tasks mainly focus on the FutureGrid resource utilization in a Cloud Computing environment, which enables elastic resource allocation. Meanwhile, we are working on a dynamic provisioning software, salsaDPI, which automates job executions running on FutureGrid Eucalyptus cloud service. As previously discussed, salsaDPI will potentially be a component of the execution broker.

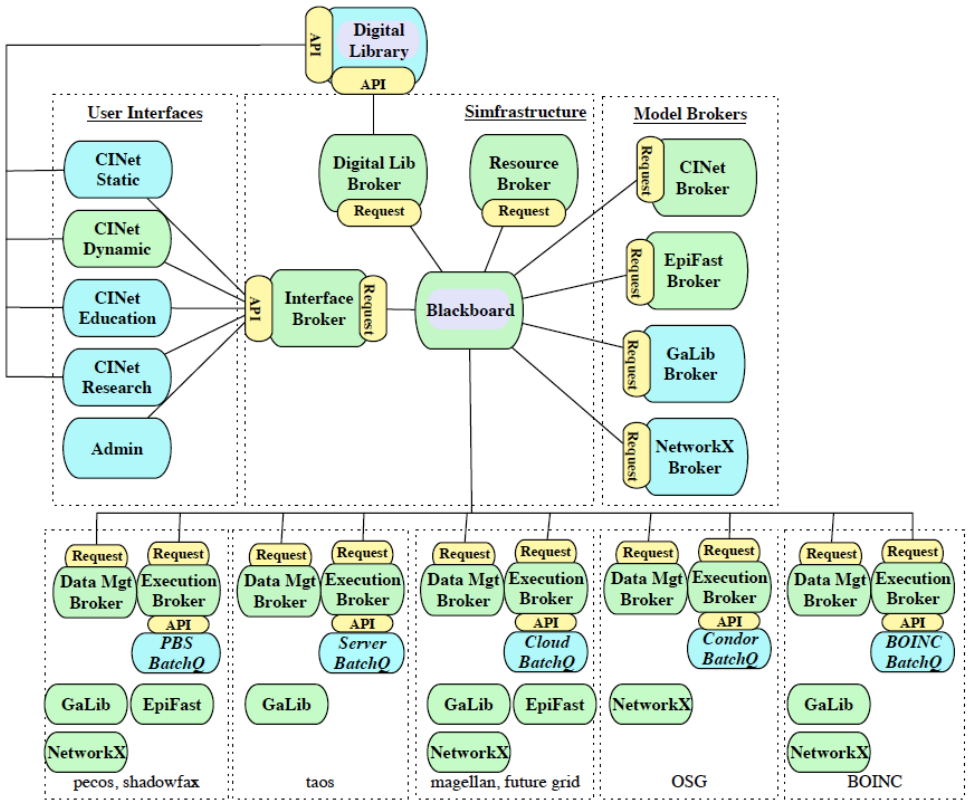


Figure 3. Current CINET Blackboard System Architecture

5. salsaDPI System Design and Implementation

Reproducibility on public/private clouds is very important for any commercial and science-executable binaries. With this reproducible feature, much data analysis could be done without worrying about the complicated system setup progress, the learning curve of different cloud infrastructure tools, and whether someone has detailed background knowledge of Cloud. Programmers or scientists can focus on writing their own applications and fire them on Cloud with the support of scalability. salsaDPI has been tested and runs user-defined binaries with more than 80 VM instances on FutureGrid Eucalyptus cloud.

The design goals of salsaDPI are:

1. Support various cloud infrastructure and permanent storage
2. Automate environment settings and application execution
3. On-demand resource allocation using virtual clusters
4. Support user-defined binary and obtain result in individual VMs
5. Support scalable analytics with Hadoop MapReduce and Twister iterative MapReduce platforms.

Automate environment settings and application execution

At a higher level, salsaDPI currently provides a configurable interface which defines resource selection, location and usage, on-demand software requirements, and user-defined binary parameters. Figure 4 shows a configuration example of Hadoop WordCount program in sandbox mode. This configuration file is written in JSON [9] format. There are five important JSON objects: *mode, chef/eucaInfo, ssh, softwareRecipes, and applicationParameters*. Once these parameters are verified by the salsaDPI driver program, the driver communicates with a hosted chef server to receive authorization and permission by calling a chef plugin, knife-eucalyptus. This plugin is involved internally from salsaDPI driver which submits compute node startup requests to the FutureGrid Eucalyptus service. Once chef server detects that the requested VMs are running, it installs the software stack according to the software recipes specified in salsaDPI configuration object on every compute node. Afterwards the salsaDPI driver deploys the user-defined program binary, e.g. Hadoop WordCount, and copies program-related files such as input and binary dependency files to each compute node. Afterwards job execution commands are sent via ssh remote call and return results back to the remote laptop/machine or a permanent storage like Walrus. Finally, driver program terminates the compute nodes and releases the resources back to the Eucalyptus service pool. The detailed control flow is illustrated in Figure 2. Note that current interface could be extended to support more features if needed.

```
{
'mode':'sandbox',
'chef':{'chefSoloRecipeUrls':'http://129.79.49.248/chef-solo.tar.gz',
'chefSoloConfFilePath':'/root/salsaDPI/solo.rb'},

'ssh':{'SSHLoginUsername':'root',
'SSHPrivateKeyPath':'/root/.ssh/id_rsa' },

'softwareRecipes':['recipe[hadoopSandbox]'],

'applicationParameters':{'applicationType':'Hadoop',
'localPathOfProgramBinary':'/root/salsaDPI/apps/hadoopWordCount.jar',
'localPathOfProgramInput':'/root/salsaDPI/input/hadoopWordCountInput.txt',
'localPathOfBinaryDependency':'',
'programExecuteLocation':'',
'programArgs':'bin/hadoop jar #_JAR_# #_HDFS_INPUTDIR_# #_HDFS_OUTPUTDIR_#'}
}
```

Figure 4. Hadoop WordCount configuration example

Figure 5 shows the architecture of salsaDPI's internal communication. From the client side, we run a java executable salsaDPI driver program which validates and converts the configuration file into an object DPIConf. This DPIConf contains information defined by the user interface. Then the Driver creates a Job Information object, JobInfo, which is used to construct the infrastructure and software level resources deployment. Based on the JobInfo, we utilize the java external process (java.lang.Process) class to call knife binary and start compute nodes with the support of an external hosted Chef Server. In between pending resources and after the compute nodes are ready for the next stage, we rely on Chef to monitor the deployment progress. Once compute nodes have been booted and installed user-defined software, all the resource information, such as hostname and IPs, is stored within this JobInfo object. Driver continues to upload the user applications and inputs to each compute node via general ssh communication (a java library Jsch). In the end, we remotely call the application execution commands and return the result back to the working client.

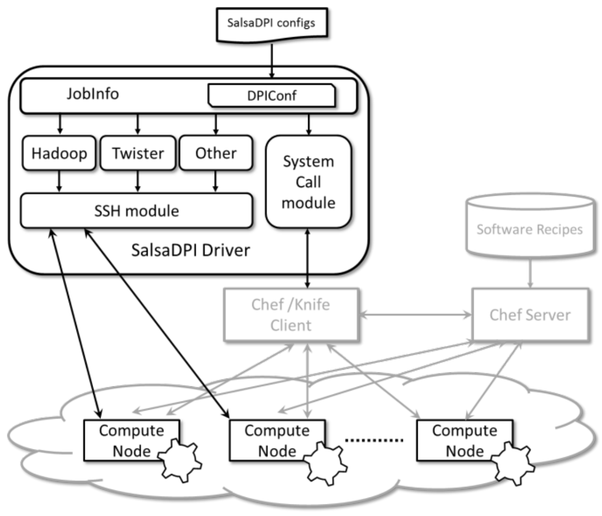


Figure 5. salsaDPI Architecture

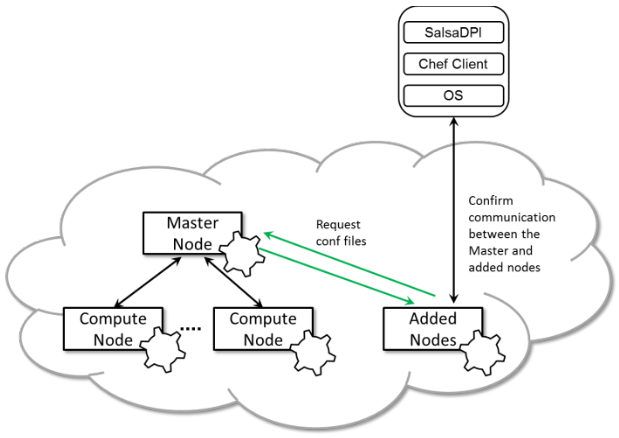


Figure 6. salsaDPI add nodes communication

Support various cloud infrastructure and permanent storage

FutureGrid Eucalyptus [10] and OpenStack [11] cloud has been tested and supported. In addition to computing resources, we have added permanent storage support such as Walrus (open source S3-like storage) and HTTP data parameters to store program-related read-only files and output.

For example, 'applicationParameters' object in the configuration file can be filled with existing online data in HTTP format. This setting is also applied for program-related dependencies to the fields of 'localPathOfProgramBinary', 'localPathOfProgramInput' and 'localPathOfProgramDB'. Figure 7 shows a Twister WordCount example which use external HTTP resources as program binary and input.

```
{   // mode = 'cloud'
    'mode':'cloud',
   // euca cloud parameters
    'eucaInfo':{'IaaS':'openstack',
                'eucarcFilePath':'/root/novarc',
                'eucaImageEmi':'4bc49f12-955b-47ce-8bd1-1933433439b2',
                'eucaSSHPublicKey':'taklwu_openstack', // replace stephen to your pub key name
                'eucaVmType':'2',
                'amountOfInstances':2},

    'ssh':{'SSHLoginUsername':'root',
          'SSHPrivateKeyPath':'/root/taklwu_openstack.pem'}, // replace stephen.pem to your private key


    'softwareRecipes':['recipe[twisterCloud]'],


    'applicationParameters':{'applicationType':'Twister',
                 'localPathOfProgramBinary':'http://salsahpc.indiana.edu/salsadpi/apps/Twister-WordCount-0.9.jar',
                'localPathOfProgramInput':'http://salsahpc.indiana.edu/salsadpi/input/twisterWordCountInput.tar.gz',
                'localPathOfProgramDB':'',
                'programExecuteLocation':'samples/wordcount/bin',
                'twisterInputFilesPreFix':'wc_data',
                'programArgs':'./run_wc.sh #_TWISTER_PARTITION_FILE_# #_TWISTER_OUTPUTDIR_#/wc.out 4 1'}
}
```

Figure 7. Twister WordCount on India OpenStack configuration example

Execute user-defined binary and obtain result

From the CINET portal, users can select or upload the program inputs according to the selected algorithms and applications. This input will be passed to the data management broker and execution broker in order to automatically run the chosen algorithm/application. This part of functions generally supports prepared CINET application library.

Partial-failure jobs

In the design of salsaDPI, a job is a continuous resource aggregation from the same IaaS resource provider. In other words, if there is a single node failure out of 10 requested VM nodes, this job is considered as failed and does not proceed with any further operations. Failures were common when there were many submitted jobs, especially when IaaS providers were running out of resources, which results in a waste of started resources. In the current release, salsaDPI allows a job to execute normally with a minimum number of started nodes, by default set to 1, although the overall execution time is slower than expected. The related evaluation results will be shown in section 8.

On-demand resource allocation

For the command-line and programmable interface, we support long running instances after execution of the user specified application. This advanced resource control is designed for applications requiring multiple-rounds of calculation, and supports HBase as persistent storage. Furthermore, the amount of running instances of existing virtual clusters can be adjusted by adding/removing VM nodes with the programmable interface cgl.salsa.salsadpi.plugin.AddANodeToExistingIaaSCluster. The user or program developer needs to provide the cluster name (normally master node hostname), master node hostname and virtual network private IP address, a shorter salsaDPI configuration file (excluding the application object), and amount of add/remove instance(s). Figure 8 shows a salsaDPI configuration file for adding HBase regionserver nodes to an existing cluster which installs HBase and Ganglia when starting the instance.

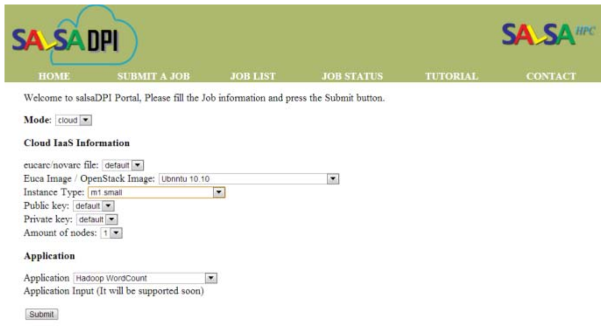
```
{
'mode':'cloud',
'eucaInfo':{ 'IaaS':'eucalyptus',
'eucarcFilePath':'/root/eucarc',
'eucaImageEmi':'emi-A8F63C29',
'eucaSSHPublicKey':'stephen',
'eucaVmType':'c1.medium'},

'ssh':{'SSHLoginUsername':'root',
'SSHPrivateKeyPath':'/root/.ssh/id_rsa' },

'softwareRecipes':['recipe[hbase]', 'recipe[ganglia]']
}
```

Figure 8. A shorter salsaDPI configuration for HBase and Ganglia

When salsaDPI receives addnodes requests, it starts the VM instances with the provided shorter salsaDPI configuration file, the same as starting a new cluster. Once the VMs have begun their work, salsaDPI sends ssh commands to the specified nodes for requesting configuration files from the Master Node. If configuration files are received correctly, user-defined services such as Ganglia, Hadoop, Twister, or HBase, start running and new started VMs are added to the cluster. Figure 6 shows the communication among salsaDPI, the master node of an existing virtual cluster, and new added nodes. salsaDPI continues tracking these ssh communications and logs errors.

salasDPI Online Interface

In addition to the command line and programmable interface, we also built a user-friendly online prototype for submitting pre-packaged applications, Hadoop/Twister WordCount and Hadoop/Twister Kmeans. The website link is http://salsahpc.indiana.edu/salsaDPI/. Note that this prototype could be extended to provide more functions, such as allowing users to upload their own applications and program input. The current online portal allows users to select several parameters like Eucalyptus and OpenStack instance type, total amount of compute nodes, and type of applications. Once the job is submitted from the portal, it executes a backend salsaDPI job on the same machine as mentioned in the above section. The submitted job is capable of being tracked from the real-time job listing page. Once finished, it provides a direct link to the program output. These features are shown in Figure 9.



Welcome to salsaDPI Portal, Please fill the Job information and press the Submit button.

Mode: cloud

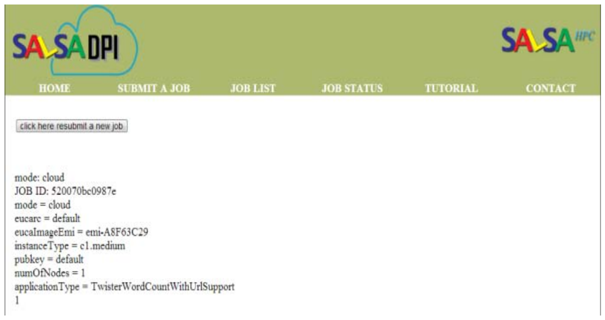
Cloud IaaS Information

eucarc/novarc file: default
Euca Image / OpenStack Image: Ubnntu 10.10
Instance Type: m1.small
Public key: default
Private key: default
Amount of nodes: 1

Application

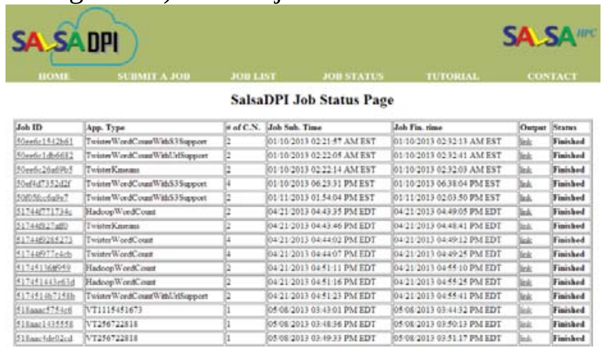
Application Hadoop WordCount
Application Input (It will be supported soon)

Figure 9 a). Online job submission interface



```
mode: cloud
JOB ID: 520070bc0987e
mode = cloud
eucarc = default
eucaImageEmi = emi-A8F63C29
instanceType = c1.medium
pubkey = default
numOfNodes = 1
applicationType = TwisterWordCountWithUrlSupport
1
```

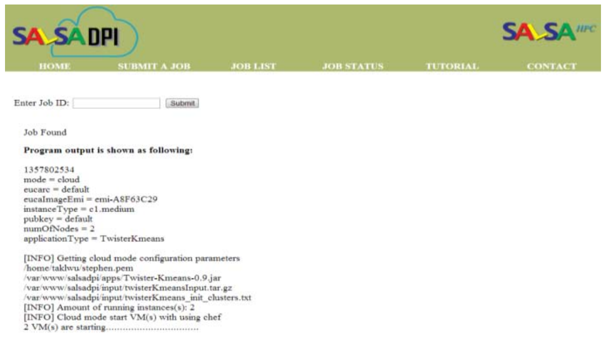
Figure 9 b). Message after a job submitted



SalsaDPI Job Status Page

Job ID	App. Type	# of C.N.	Job Sub. Time	Job Fin. rime	Output	Status
50ee6c1542b61	TwisterWordCountWithS3Support	2	01/10/2013 02:21:57 AM EST	01/10/2013 02:32:13 AM EST	link	Finished
50ee6c1db6682	TwisterWordCountWithUrlSupport	2	01/10/2013 02:22:05 AM EST	01/10/2013 02:32:41 AM EST	link	Finished
50ee6c26a69b5	TwisterKmeans	2	01/10/2013 02:22:14 AM EST	01/10/2013 02:32:03 AM EST	link	Finished
50ef4d7352d2f	TwisterWordCountWithS3Support	4	01/10/2013 06:23:31 PM EST	01/10/2013 06:38:04 PM EST	link	Finished
50f05fcc6a9e7	TwisterWordCountWithS3Support	2	01/11/2013 01:54:04 PM EST	01/11/2013 02:03:50 PM EST	link	Finished
51744f771734c	HadoopWordCount	2	04/21/2013 04:43:35 PM EDT	04/21/2013 04:49:05 PM EDT	link	Finished
51744f827aff0	TwisterKmeans	2	04/21/2013 04:43:46 PM EDT	04/21/2013 04:48:41 PM EDT	link	Finished
51744f9285273	TwisterWordCount	4	04/21/2013 04:44:02 PM EDT	04/21/2013 04:49:12 PM EDT	link	Finished
51744f977e4cb	TwisterWordCount	4	04/21/2013 04:44:07 PM EDT	04/21/2013 04:49:25 PM EDT	link	Finished
51745136ff959	HadoopWordCount	2	04/21/2013 04:51:11 PM EDT	04/21/2013 04:55:10 PM EDT	link	Finished
517451443e63d	HadoopWordCount	2	04/21/2013 04:51:16 PM EDT	04/21/2013 04:55:25 PM EDT	link	Finished
5174514b7158b	TwisterWordCountWithUrlSupport	2	04/21/2013 04:51:23 PM EDT	04/21/2013 04:55:41 PM EDT	link	Finished
518aaac5754c6	VT1115451673	1	05/08/2013 03:43:01 PM EDT	05/08/2013 03:44:32 PM EDT	link	Finished
518aac1435558	VT256722818	1	05/08/2013 03:48:36 PM EDT	05/08/2013 03:50:13 PM EDT	link	Finished
518aac4de02cd	VT256722818	1	05/08/2013 03:49:33 PM EDT	05/08/2013 03:51:17 PM EDT	link	Finished

Figure 9 c). Job status page



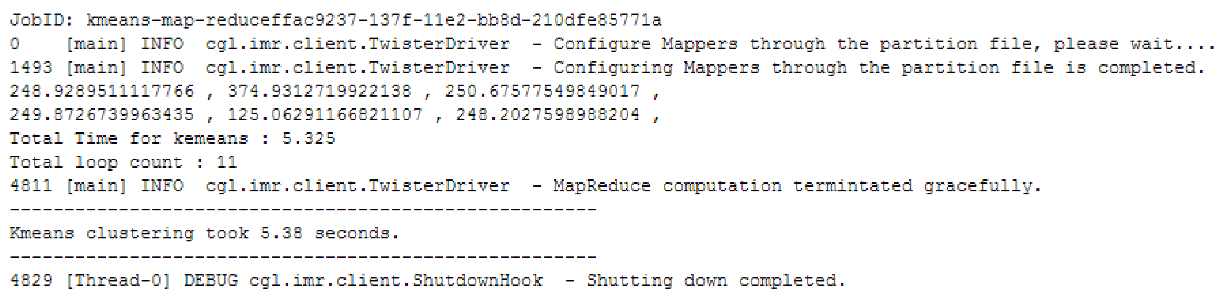
Enter Job ID:

Job Found

Program output is shown as following:

```
1357802534
mode = cloud
eucarc = default
eucaImageEmi = emi-A8F63C29
instanceType = c1.medium
pubkey = default
numOfNodes = 2
applicationType = TwisterKmeans

[INFO] Getting cloud mode configuration parameters
/home/taklwu/stephen.pem
/var/www/salsadpi/apps/Twister-Kmeans-0.9.jar
/var/www/salsadpi/input/twisterKmeansInput.tar.gz
/var/www/salsadpi/input/twisterKmeans_init_clusters.txt
[INFO] Amount of running instances(s): 2
[INFO] Cloud mode start VM(s) with using chef
2 VM(s) are starting.................................
```

Figure 9 d). Real time job tracking page



```
JobID: kmeans-map-reduceffac9237-137f-11e2-bb8d-210dfe85771a
0    [main] INFO  cgl.imr.client.TwisterDriver  - Configure Mappers through the partition file, please wait....
1493 [main] INFO  cgl.imr.client.TwisterDriver  - Configuring Mappers through the partition file is completed.
248.9289511117766 , 374.9312719922138 , 250.67577549849017 ,
249.8726739963435 , 125.0629116621107 , 248.2027598988204 ,
Total Time for kemeans : 5.325
Total loop count : 11
4811 [main] INFO  cgl.imr.client.TwisterDriver  - MapReduce computation termintated gracefully.
------------------------------------------------------
Kmeans clustering took 5.38 seconds.
------------------------------------------------------
4829 [Thread-0] DEBUG cgl.imr.client.ShutdownHook  - Shutting down completed.
```

Figure 9 e). Example of Twister Kmeans output

6. salsaScaler: A resource monitoring interface

In addition to salsaDPI, we also provide a monitoring software package, salsaScaler, to monitor the activated virtual cluster system-level and service-level resources. Figure 10 shows control flow for salsaScaler. On each started instance, we install and run Ganglia [12] resource monitoring tool as foundational software, where only the first/master node runs the gmetad [13] daemon to collect node system usages monitored by gmond [13]. salsaScaler communicates with known gmetad daemons among the virtual clusters and analyzes the collected performance metric based on the user-defined interests. It collects performance metrics from each cluster every 10 seconds and summarizes a general decision when four different metrics are received. Based on the decision, salsaScaler notifies salsaDPI to add nodes or to remove them from a monitored cluster. Users of salsaScaler must write a metric.xml configuration file to specify the targeted cluster with listed hostname and public IP address, the metrics interests, and SLA calculation method. Figure 11 shows an example for monitoring two clusters while watching the average usages on cpu_idle, cpu_aidle, mem_cached, and swap_free. In addition, we also provide a command-line interface to manually add targeted clusters, add nodes to an existing cluster and list information of all the current monitored instances.

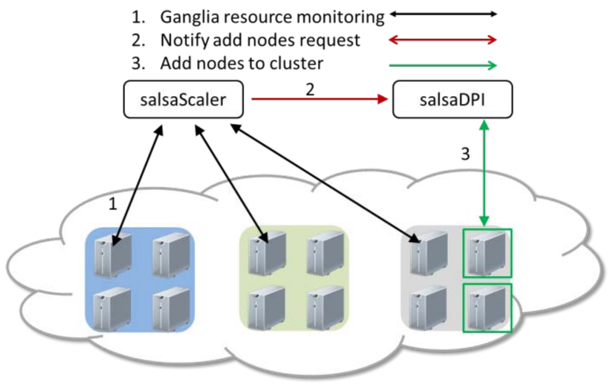


Figure 10. salsaScaler Architecture

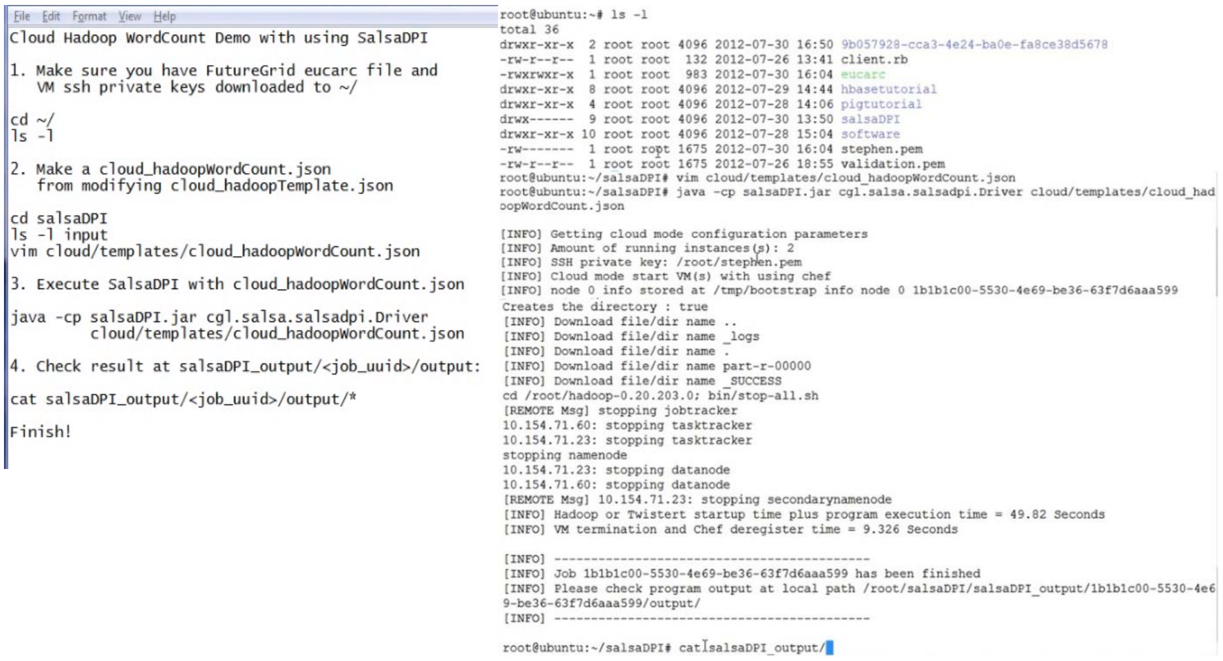
```
<!DOCTYPE properties SYSTEM
"http://java.sun.com/dtd/properties.dtd">
<configuration>
<property>
<name>targeted.clusters</name>
<value>LocalPlayGround:129.79.49.181, i-
07DF4398:149.165.158.203</value>
<description>specify the your own targeted headnode which
has installed gmetad</description>
</property>
<property>
<name>metrics.interests</name>
<value>cpu_idle, cpu_aidle, mem_cached, swap_free</value>
<description>specify the your own metric
interests</description>
</property>
<property>
<name>sla.calculation.method</name>
<value>average</value>
</property>
</configuration>
```

Figure 11. salsaScaler configuration file

7. Use Cases of salsaDPI

In the summer of 2012, Indiana University and 9 other sites of university level (http://www.vscse.org/summerschool/2012/scss.html) held a virtual conference across nations. This conference mainly introduced cloud technologies to graduate-level students and staff. A salsaDPI online tutorial page is located at http://salsahpc.indiana.edu/ScienceCloud/reproduce-intro.html, one of the main tutorials of this activity. Figure 12 is one of the photos taken during this event. It shows the ability to deploy single node and virtual runtime environments for Hadoop and Twister applications from a raw OS environment (without any software pre-installed). Also, it automatically executes Hadoop and Twister applications such as WordCount and Kmeans after installing the selected software stack (see Figure 4 for configuration details). Figure 13 shows snapshots when salsaDPI is running with a user-defined Hadoop WordCount configuration file (the video link can be seen at http://www.youtube.com/watch?feature=player_embedded&v=kWom0lj8qxI, please view it with video quality of 1080p).

Figure 12. Science Cloud Summer School 2012



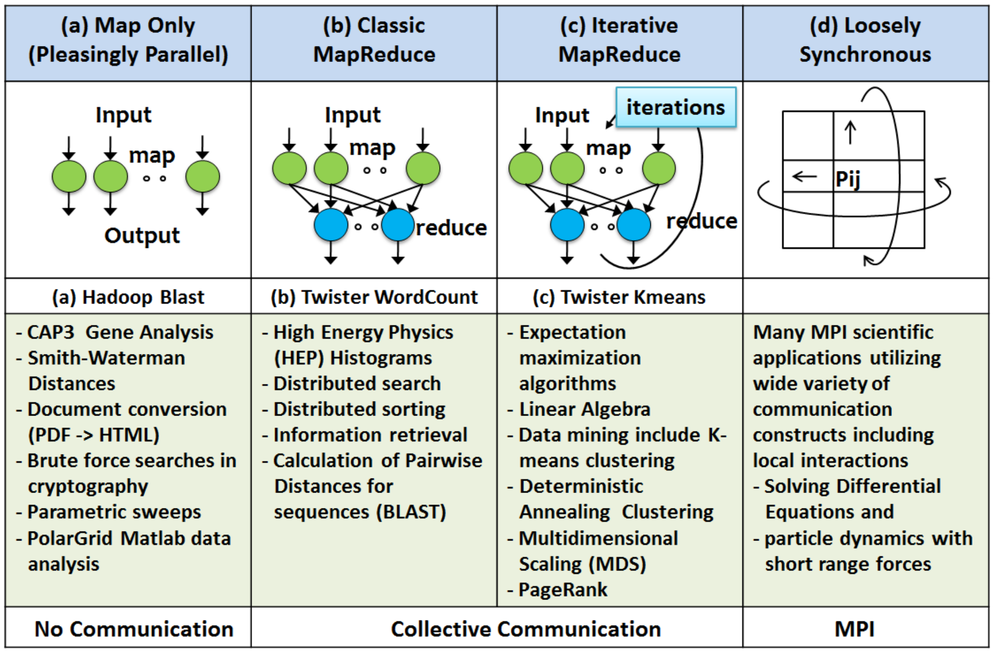
```
Cloud Hadoop WordCount Demo with using SalsaDPI

1. Make sure you have FutureGrid eucarc file and
   VM ssh private keys downloaded to ~/

cd ~/
ls -l

2. Make a cloud_hadoopWordCount.json
   from modifying cloud_hadoopTemplate.json

cd salsaDPI
ls -l input
vim cloud/templates/cloud_hadoopWordCount.json

3. Execute SalsaDPI with cloud_hadoopWordCount.json

java -cp salsaDPI.jar cgl.salsa.salsadpi.Driver
        cloud/templates/cloud_hadoopWordCount.json

4. Check result at salsaDPI_output/<job_uuid>/output:

cat salsaDPI_output/<job_uuid>/output/*

Finish!
```

```
root@ubuntu:~# ls -l
total 36
drwxr-xr-x  2 root root 4096 2012-07-30 16:50 9b057928-cca3-4e24-ba0e-fa8ce38d5678
-rw-r--r--  1 root root  132 2012-07-26 13:41 client.rb
-rwxrwxr-x  1 root root  983 2012-07-30 16:04 eucarc
drwxr-xr-x  8 root root 4096 2012-07-29 14:44 hbasetutorial
drwxr-xr-x  4 root root 4096 2012-07-28 14:06 pigtutorial
drwx------  9 root root 4096 2012-07-30 13:50 salsaDPI
drwxr-xr-x 10 root root 4096 2012-07-28 15:04 software
-rw-------  1 root root 1675 2012-07-30 16:04 stephen.pem
-rw-r--r--  1 root root 1675 2012-07-26 18:55 validation.pem
root@ubuntu:~/salsaDPI# vim cloud/templates/cloud_hadoopWordCount.json
root@ubuntu:~/salsaDPI# java -cp salsaDPI.jar cgl.salsa.salsadpi.Driver cloud/templates/cloud_had
oopWordCount.json

[INFO] Getting cloud mode configuration parameters
[INFO] Amount of running instances(s): 2
[INFO] SSH private key: /root/stephen.pem
[INFO] Cloud mode start VM(s) with using chef
[INFO] node 0 info stored at /tmp/bootstrap info node 0 1b1b1c00-5530-4e69-be36-63f7d6aaa599
Creates the directory : true
[INFO] Download file/dir name ..
[INFO] Download file/dir name _logs
[INFO] Download file/dir name .
[INFO] Download file/dir name part-r-00000
[INFO] Download file/dir name _SUCCESS
cd /root/hadoop-0.20.203.0; bin/stop-all.sh
[REMOTE Msg] stopping jobtracker
10.154.71.60: stopping tasktracker
10.154.71.23: stopping tasktracker
stopping namenode
10.154.71.23: stopping datanode
10.154.71.60: stopping datanode
[REMOTE Msg] 10.154.71.23: stopping secondarynamenode
[INFO] Hadoop or Twistert startup time plus program execution time = 49.82 Seconds
[INFO] VM termination and Chef deregister time = 9.326 Seconds

[INFO] ------------------------------------------
[INFO] Job 1b1b1c00-5530-4e69-be36-63f7d6aaa599 has been finished
[INFO] Please check program output at local path /root/salsaDPI/salsaDPI_output/1b1b1c00-5530-4e6
9-be36-63f7d6aaa599/output/
[INFO] ------------------------------------------

root@ubuntu:~/salsaDPI# catIsalsaDPI_output/
```

Figure 13. Running salsaDPI Hadoop WordCount on FutureGrid

8. Evaluation

We evaluate systematically different applications. Here, (a) Hadoop BLAST represents the pleasingly parallel program; (b) Hadoop/Twister WordCount exemplifies classic MapReduce programming model; (c) Twister Kmeans stands for iterative applications. We ran performance tests on two different resources – FutureGrid Eucalyptus and OpenStack Cloud.



(a) Map Only (Pleasingly Parallel)	(b) Classic MapReduce	(c) Iterative MapReduce	(d) Loosely Synchronous
Input map Output	Input map reduce	Input iterations map reduce	Pij
(a) Hadoop Blast	(b) Twister WordCount	(c) Twister Kmeans	
- CAP3 Gene Analysis - Smith-Waterman Distances - Document conversion (PDF -> HTML) - Brute force searches in cryptography - Parametric sweeps - PolarGrid Matlab data analysis	- High Energy Physics (HEP) Histograms - Distributed search - Distributed sorting - Information retrieval - Calculation of Pairwise Distances for sequences (BLAST)	- Expectation maximization algorithms - Linear Algebra - Data mining include K-means clustering - Deterministic Annealing Clustering - Multidimensional Scaling (MDS) - PageRank	Many MPI scientific applications utilizing wide variety of communication constructs including local interactions - Solving Differential Equations and - particle dynamics with short range forces
No Communication	Collective Communication		MPI

Figure 14. Classification of Applications and Communication Patterns

(a) Hadoop BLAST

We run Hadoop BLAST tests on both Euclyptus and Openstack Cloud. The result in Figure 15 indicates that partial failure jobs had a longer startup time and execution time. In the worse case, 2 out of 4 nodes failed, and the Hadoop BLAST computation was postponed for a 600 second timeout from the VM response. Due to this, execution time doubles, as it only obtained half of the requested computing resources. Virtual instances may fail to start due to lack of resources and IaaS DoS prevention setting.

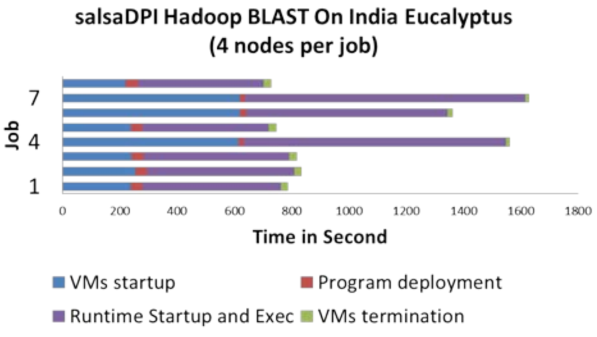


Figure 15a. Hadoop Blast on Euca

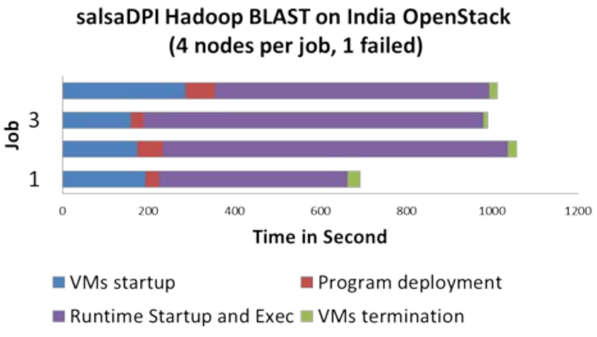


Figure 15b. Hadoop Blast on Openstack

(b) Twister WordCount

Twister WordCount application runs on two c1.medium eucalyptus instances. It is periodically submitted (every 3 seconds) by running a salsaDPI client from an Ubuntu 12.04 machine located within the same intranet. This setup guarantees the chef server and salsaDPI client have seamless connection as they both are linked to the same Gigabit network with low latency. We capture the VM startup time, program data and dependencies deployment time, runtime startup and program execution time, and finally VM termination time. Initially we executed 40 jobs to determine the feasibility, then another 60 jobs to overload the system capacity. As shown in figure 16, VM startup and termination time are the two largest portions. Also, their huge variation is due to connection latencies from the India Eucalyptus headnode when it handles a large amount of connections and bootstrap VMs onto physical machines. We verify the slow response time by running the 60 jobs test. 1 out of 40 jobs and 7 out of 60 jobs failed in our test cases. The reasons are mainly related to VM image deployment on physical machines and the incoming connection capacity of India Eucalyptus headnode. Results are shown in Figure 17.

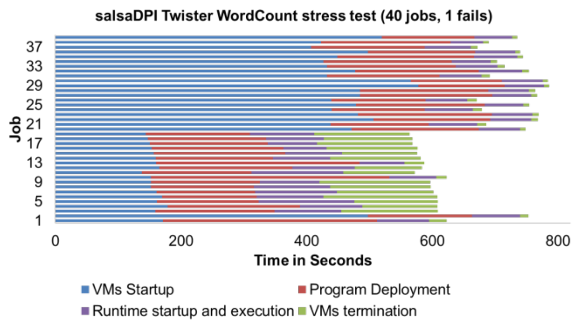


Figure 16a. 40 Twister WordCount jobs on India Eucalyptus

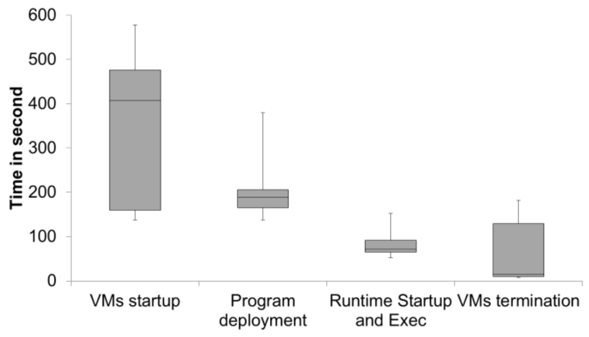


Figure 16b. Detail deployment time

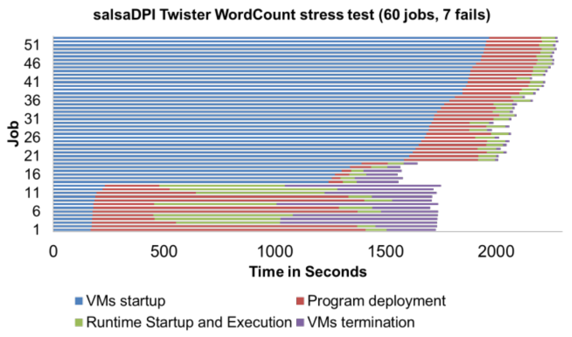


Figure 17a. Twister WordCount on India Eucalyptus

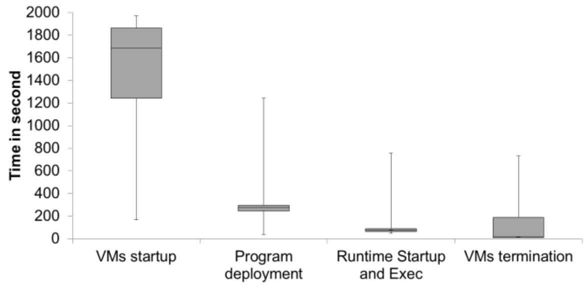


Figure 17b. Detail deployment time

In addition to Inda with Eucalyptus, the same tests were also run on India and Sierra with OpenStack. However, due to different system settings, we cannot obtain the same large-scale results as on OpenStack. For every job, Twister WordCount application runs on two m1.small openstack instances periodically submitted (every 30 seconds) by running a salsaDPI client from the same Ubuntu 12.04 machine. The 30-second delay is decided by system setting, when OpenStack only allows 10 POST requests per minute to their RESTful API. Moreover, India OpenStack has a special automatic floating IP assignment which conflicts with the chef knife-openstack plugin used by salsaDPI Driver. This floating IP conflict may be the critical factor which limits the amount of running instances. Figure 18 shows a 5-jobs test result run on India-OpenStack. The VM startup time in OpenStack is much faster, despite our test size being so small.

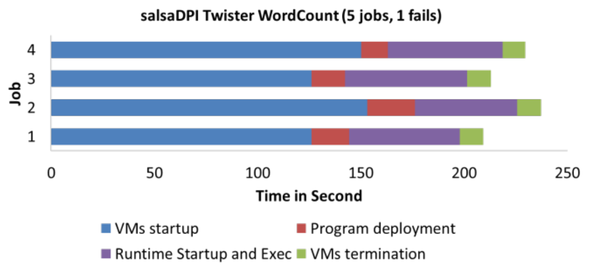


Figure 18a. Twister WordCount on India OpenStack

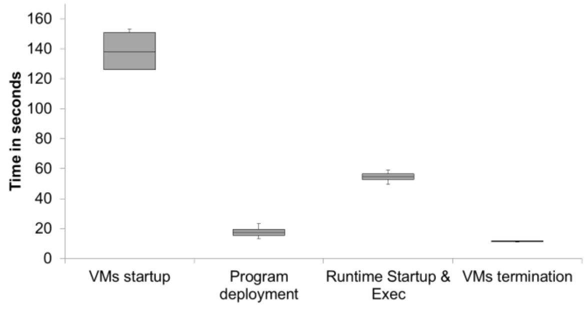


Figure 18b. Detail deployment time

(c) Hadoop and Twister Kmeans

For Hadoop Kmeans, the input data size is 50,000 data points with set 500 centroids, and the program runs for 10 iterations. For Twister Kmeans, the input data size is 80,000 data points with set 2 initial centroids. The program stops if the centroids errors are the same as the previous iteration. As shown in Figure 19, Hadoop Kmeans and Twister Kmeans run without any failure on India Eucalyptus.

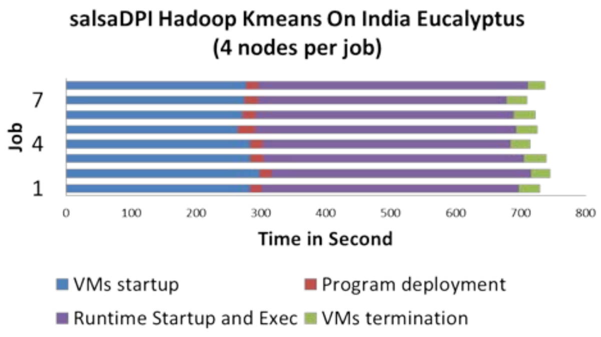


Figure 19a Hadoop Kmeans on Euca

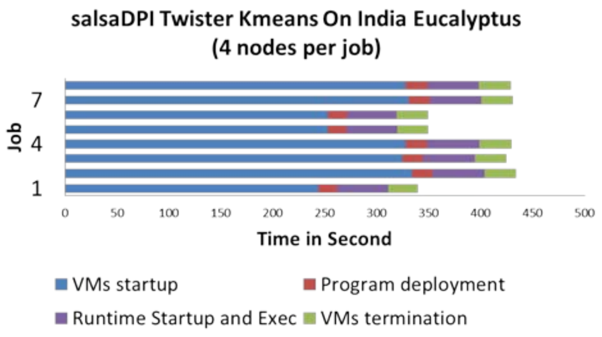


Figure 19b. Twister Kmeans on Euca

The same input datasets are used to run tests on India OpenStack. However, due to the 10 Posts limitation on India OpenStack, we got a different result, as can be seen in Figure 20. A total of 5 jobs were submitted and the break interval was 30 seconds. Both tests had 1 failure job and partial failure nodes. The startup time of Hadoop Kmeans is better due to it having less nodes failed for each node.

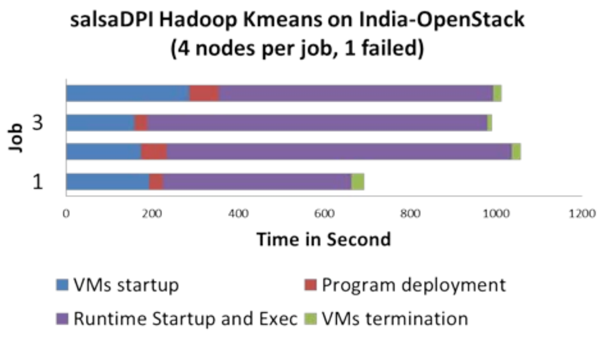


Figure 20a. Hadoop Kmeans on Openstack

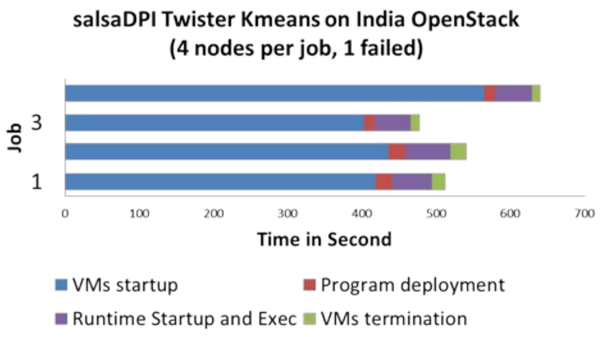


Figure 20b. Twister Kmeans on Openstack

9. Conclusion and Future Works

We at Indiana University have delivered the command-line runnable, programmable, and user-friendly initial framework salsaDPI to the CINET project, which shows the reproducibility in dynamic computing environments. In addition, there are possible future enterprises to raise this collaboration to an advanced level of research.

Statement of Work

We list future work (after September 18th, 2013) below:

- Continue salsaDPI integration with VT team.
- For salsaScaler, we need to identify the most importance performance metrics and find analytical algorithms to define performance leaks on a running cluster. These parts need more research work and more advice from Prof. Judy Qiu.
- Improve salsaScaler to provide a programmable interface to calculate collected metrics.

- Replace execution steps from knife-euca/knife-openstack to start VMs using jclouds APIs and purely Java SSH sessions. This contribution makes salsaDPI portable and platform-free without installing any dependency-like chef-client and the rest of the knife-related APIs.

Acknowledgments

We would like to thank Dr. Madhav Marathe, Dr. Keith Bisset, and Hemanth Makkapati from the VT team at Virginia Tech. Our thanks also to Dr. Geoffrey Fox from IU team at Indiana University, and the FutureGrid support team, especially Dr. Javier Diaz and system admin Koji Tanaka. This project is in part supported by NSF Grant OCI-1032677.

Collaboration with VT Team

This has been a collaboration project with the Virginia Tech (VT) team since early 2012. This section illustrates milestones that have been reached and those that will be achieved in the future.

Work before 2013

The salsahpc group at Indiana University Bloomington has been providing assistance to the Virginia Tech team on interfacing with FutureGrid computing resources, virtual machine environment setups and providing salsaDPI. This portable API has been successfully deployed and used by 300 nationwide graduate-level students and IT staff at Science Cloud Summer School 2012 [7]. Their website can be seen at https://portal.futuregrid.org/projects/241. Details about this summer school activity are included in later sections. Moreover, a programming interface will be ready in mid-January which can be integrated with CINET software package.

Work between January and September 2013

The following improvements have been made for salsaDPI between January 1st, 2013 and September 18th, 2013:

- Support India-OpenStack and HTTP input
- Web Interface reconstruction
- A heavy-duty test on Eucalyptus and OpenStack
- Support partial failure job
- Provide on-demand resource extension/reduction to existing user-level clusters
- Provide salsaScaler for resource monitoring with Ganglia

Reference

1. *FutureGrid*. Available from: https://portal.futuregrid.org/.
2. Klinginsmith, J., M. Mahoui, and Y.M. Wu, *Towards Reproducible eScience in the Cloud*, in *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*. 2011, IEEE Computer Society. pages. 582-586. DOI: 10.1109/CloudCom.2011.89.
3. Bresnahan, J., T. Freeman, D. LaBissoniere, and K. Keahey, *Managing appliance launches in infrastructure clouds*, in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*. 2011, ACM. Salt Lake City, Utah. pages. 1-7. DOI: 10.1145/2016741.2016755.
4. *Opscode Chef*. Available from: http://www.opscode.com/chef/.
5. Fischer, J., R. Majumdar, and S. Esmaeilsabzali, *Engage: a deployment management system.* SIGPLAN Not., 2012. 47(6): p. 263-274. DOI:10.1145/2345156.2254096
6. Keahey, K., P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau, *Infrastructure Outsourcing in Multi-Cloud Environment*, in *the 8th Open Cirrus Summit*. 2012. San Jose, CA.
7. *Science Cloud Summer School 2012*. 2012 Available from: https://portal.futuregrid.org/projects/241.
8. *Oracle VM VirtualBox*. Available from: https://www.virtualbox.org/.
9. *JSON metadata format*. Available from: http://www.json.org/.
10. *Eucalyptus Open Source Cloud Software*. Available from: http://open.eucalyptus.com/.
11. Open Stack. *Open source, Open standards Cloud*. 2010 [accessed 2010 November 7]; Available from: http://openstack.org/index.php.
12. Massie, M.L., B.N. Chun, and D.E. Culler, *The Ganglia Distributed Monitoring System: Design, Implementation And Experience.* Parallel Computing, 2003. 30(Massie03theganglia): p. 2004.
13. *Ganglia Monitoring System*. Available from: http://ganglia.sourceforge.net/.