

salsaDPI: a dynamic provisioning interface for IaaS cloud

Tak-Lon (Stephen) Wu
Computer Science, School of Informatics and Computing
Indiana University, Bloomington, IN
taklwu@indiana.edu

Abstract

On-demand private cloud environment has become popular due to its flexibility and scalability. However, the complicated setup and installation process to obtain the computing resource has deterred users away. Inspired by the batch job model, we introduce a dynamic provisioning software, salsaDPI, to simplify these complications which allows domain scientists to focus on their scientific problems and run their applications on clouds with dynamic and automated installation and configurations.

1. Introduction

Batch job has been widely adopted by scientists and commercial companies to handle daily tasks that analyze terabyte of data on different computing resources. These tasks execute well in traditional static environment such as high performance cluster. However, it is not easy to run them in a dynamic environment such as 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 a urgent deployment request of computing resources. But before users submit a request to startup 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 of 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, to releasing the computing resources. Based on these considerations, we develop an experimental on-demand dynamic provisioning software, hereafter written as salsaDPI, which automates the job executions running on FutureGrid Eucalyptus cloud service.

This paper is structured as follows. Section 2 gives an overview of dynamic resource provisioning and management technologies. Section 3 discusses the related works. Section 4 describes the system design and architecture of SalsaDPI. Section 5 presents an educational use case. Section 6 draws the conclusion and discusses the future directions.

2. Background

Infrastructure-as-a-service is the basic service model of cloud computing, it provides scalable computing environment with the support of a large amount of virtual machines managed by 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 with 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 tools that deploys cloud resources and user-specified applications throughout different IaaS cloud and physical cluster automatically. It is an open source system written in Ruby, which makes 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-eucalyptus utilizes 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 communicate with cloud infrastructure API, meanwhile, Chef::Knife::Bootstrap takes chef-client installation task by loading default or user-defined ruby template 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 command.

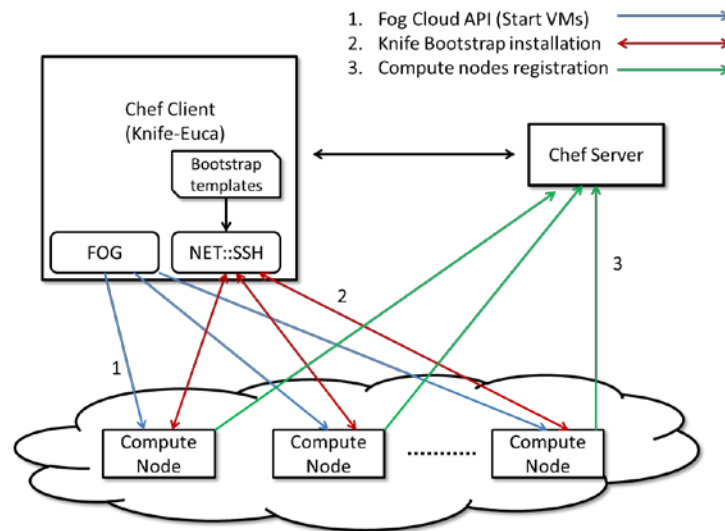


Figure 1. Chef knife-eucalyptus internal communication

3. Related Work

Engage [5] is deployment management system targets on the system deployment within a distributed environment. It defines three key components of their system: a domain-specific component describes inter-dependent software dependencies; a constraint-based component constructs the software installation plan based on the inter-dependent 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 focus on definition of resource type, software dependencies, 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 a set of IaaS cloud. Cloudinit.d is similar to Engage system, but it provide 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 failure in a distributed environment.

J. Klinginsmith [2] introduces a reproducible framework which developed based on FutureGrid and Amazon Cloud. As data locality is one of the key features of Cloud Computing, his works mainly focuses on moving computational infrastructure as close as the location of data in a dynamically deployed environment; his goals is to help users, especially eScience scientists, to repeat their daily experiments by going through two processes, which constructs on-demand infrastructure level resources and installs selected software/runtime on-the-fly when resources are booting up. Also, he writes a customized Chef Plugin (with Chef 0.10+) to read different configuration parameter in order to support various cloud IaaS API.

In addition, 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 which is distributed to 200 students of this event.

4. SalsaDPI

Based on our experience using and running different IaaS on FutureGrid, we have developed an initial framework, SalsaDPI, to automate the scientific data analysis applications of CINET project. Users of this framework can easily prepared their one-click configuration plan and launch their applications.

4.1. Overview of System Design

As shown in figure 2, requests sent to FutureGrid data management and execution brokers involve jobs running on FutureGrid Eucalyptus cloud service. Currently, we provide support and advice to CINET project from Virginian Tech about system design and test several potential components which may be integrated into data management and execution brokers. Mainly, these tasks focus on the FutureGrid resources utilization in a Cloud Computing fashion which enables elastic resource allocation. Meanwhile, we are working on an on-demand dynamic provisioning software, SalsaDPI, which automates the job executions running on FutureGrid Eucalyptus cloud service. As previously discussed, SalsaDPI will potentially be a component of the execution broker.

4.2. Features of salsaDPI

Reproducibility environment on public/private cloud is very important for any commercial and science executable binaries. With this reproducible feature, many data analysis could be done without worrying the complicated system setup progress, the learning curve of different cloud infrastructure tools, and detail background knowledge of Cloud. Programmers or Scientists can focus on writing their own applications and fires them on Cloud with the support of scalability. salsaDPI, an on-demand dynamic provisioning software runs on public/private cloud, has been tested and runs user-defined binaries with more than 80 VM instances on FutureGrid Eucalyptus cloud.

4.2.1. Design Goals

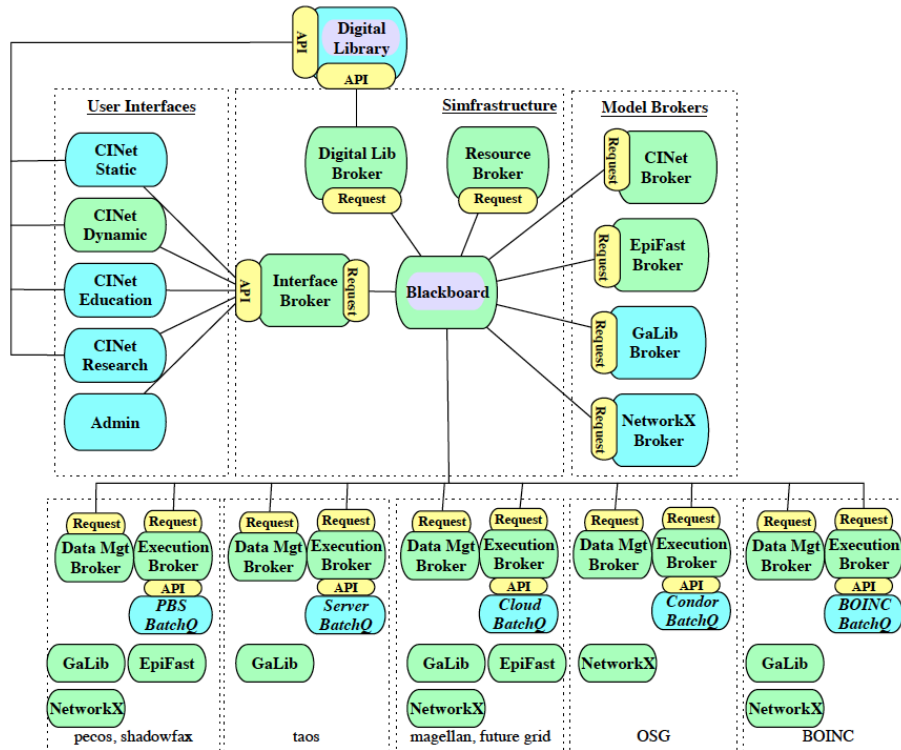


Figure 2. Current CINET Blackboard System Architecture

1. Automate environment settings and application execution
2. Support various cloud infrastructure and permanent storage
3. Execute user-defined binary and obtain result

4.2.2. System Overview

Each design goal will be explained with a description as following.

Automate environment settings and application execution

salsaDPI currently provides a configurable interface which defines resource selection and location, resource usage, on-demand software requirements, and user-defined binary's aparameters. Figure 3 shows a configuration example of Hadoop wordcount program in sandbox mode. This configuration file is written in JSON [9] format. There are four important JSON objects: mode, chef/eucalInfo, ssh, softwareRecipes, and applicationParameters. Once these parameters are verified by the salsaDPI driver program, such as checking the existence of ssh key file and program executable file, the driver communicates with a hosted chef server to get authorized and permission by calling a chef plugin, knife-eucalyptus. This plugin is involved internally from salsaDPI driver which submits computes nodes startup request to the FutureGrid Eucalyptus service. Once chef server detects the requested VMs are running, it installs the software stack according to the software recipes specified in SalsaDPI configuration object on every computes node. After the software are installed, the SalsaDPI driver deploys the user-defined program binary, e.g. Hadoop wordcount, and copy the program related files such as input files and binary dependency files to each compute node. Then, program execution commands are sent via ssh remote call and obtain result back the remote laptop/machine or a permanent storage like Walrus. Finally, driver program terminates the computes nodes and

release the resources back to the Eucalyptus service pool. The detail control flow can be seen in Figure 4. Noted 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 3. Hadoop wordcount configuration example

Figure 5 shows the architecture of SalsaDPI internal communication. From the client side, we run a java executable, SalsaDPI driver program, which validates and converts the configuration file into an object DPICConf. This DPICConf 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 utilizes the java external process (java.lang.Process) class to call knife binary and starts compute nodes with the support of an external hosted Chef Server. In between pending the resources and the compute nodes is ready to the next stage, we rely on Chef to monitor the deployment progress. Once compute nodes has been booted and been installed user-defined software, all the resources information such as hostname and IPs are 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), and at the end, we remotely call the application execution commands and obtain the result back to the working client. Figure 6 presents the detail UML of the driver program.

Support various cloud infrastructure and permanent storage

FutureGrid Eucalyptus cloud has been tested and supported, and we aims to support different cloud infrastructures such as OpenStack [10] and Nimbus [11] cloud deployed on FutureGrid in next stage. In addition to computing resources, we have already added permanent storage support such as Walrus (Open source S3-like storage) to store program related read-only files and output.

Execute user-defined binary and obtain result

From the web portal, user can select or upload the program inputs according to the selected algorithms and applications. These inputs will be passed to the data management broker and execution broker in order to automatically run the chosen algorithm/application.

4.3. System Architecture

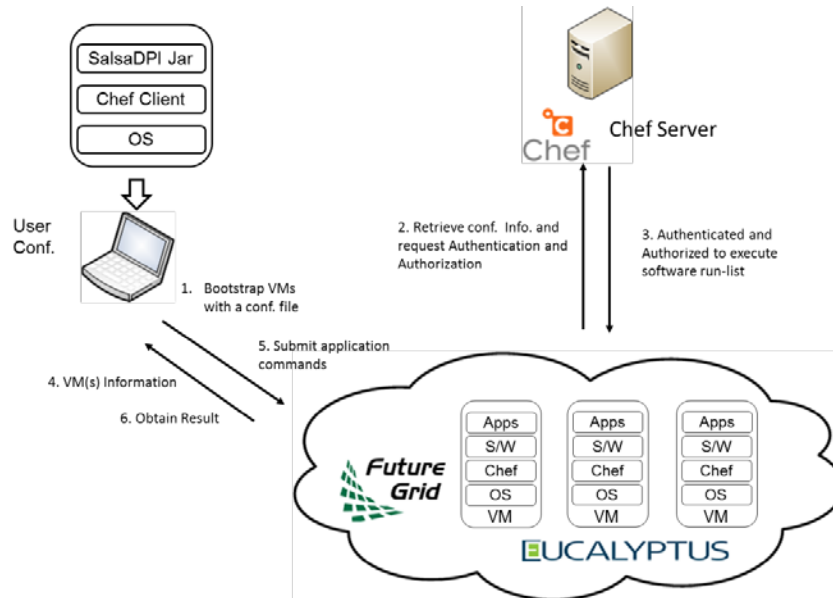


Figure 4. SalsaDPI High level control flow

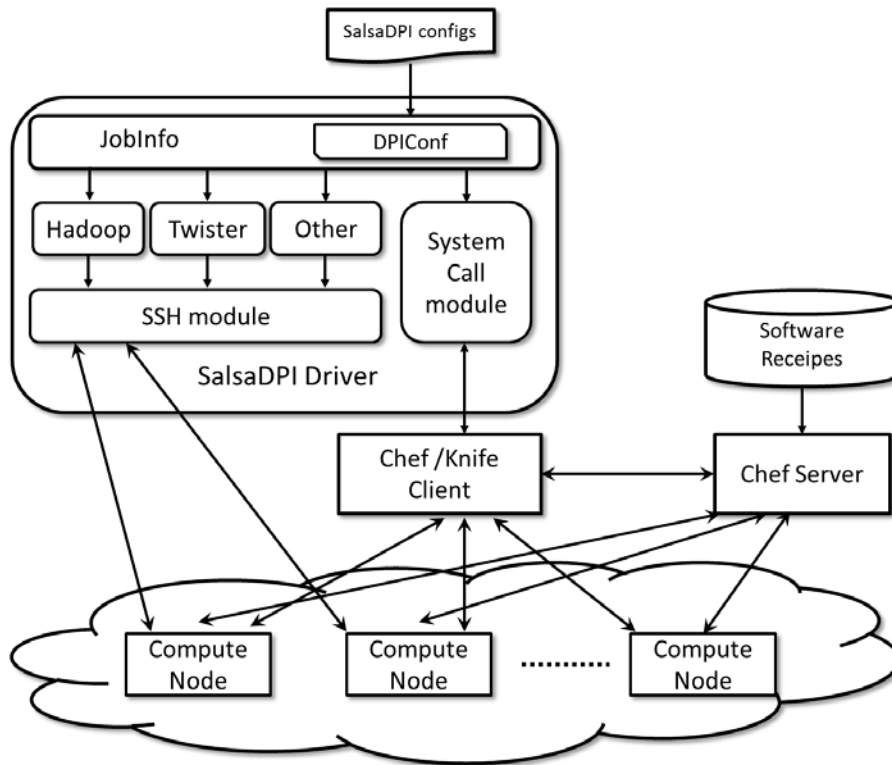


Figure 5. SalsaDPI architecture

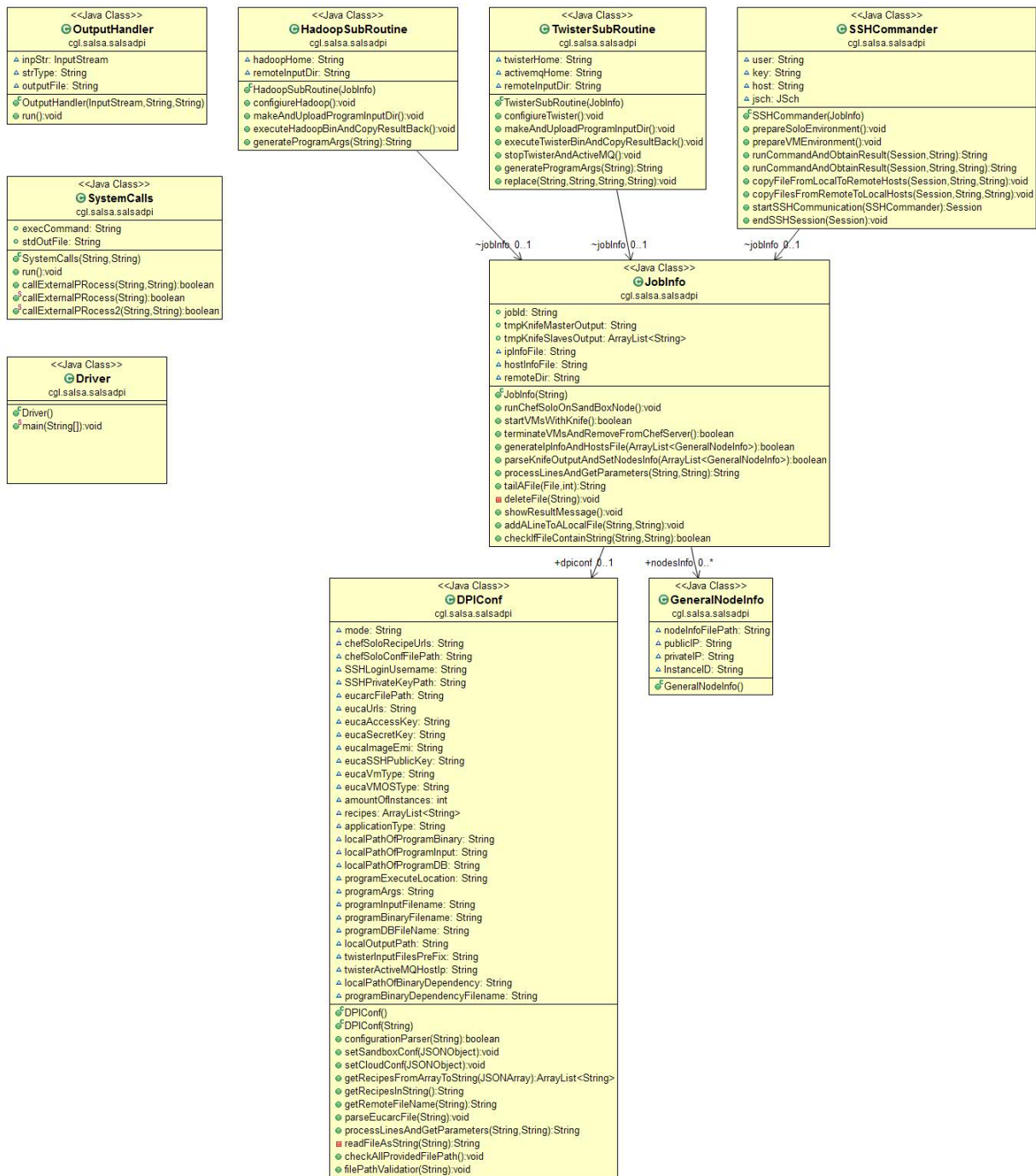


Figure 6. SalsaDPI alpha UML

4.4. User Interface

In addition to the command line 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/>. Noted that this prototype could be extended to provide more functions such as allow user upload their own applications and program input. The current online portal allows users select several parameters such as

Eucalyptus 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. Submitted Job is traceable from the real-time job list page. After the job is finished, it provides a direct link to the program output. These features are shown in Figure 7.

The screenshot shows the 'salsaDPI Portal' interface. On the left is a navigation menu with 'Home', 'List Jobs', 'Job Status', and 'About'. The main content area contains a welcome message and a form for job submission. The form includes a 'Mode' dropdown set to 'cloud'. Under 'Eucalyptus Information', there are dropdowns for 'eucarc file' (default), 'eucaImageEmi' (emi-A8F63C29), 'Instance Type' (c1.medium), 'Public key' (default), and 'Private key' (default). A text input for 'Amount of nodes' is set to '2'. Under 'Application', there is a dropdown for 'Application' (Hadoop WordCount) and a text input for 'Application Input (It will be supported soon)'. A 'Submit' button is at the bottom of the form. The footer shows the Indiana University Bloomington logo.

Figure 7 a). Online job submission interface

The screenshot shows the 'salsaDPI Portal' interface after a job submission. The navigation menu is the same. The main content area now displays a confirmation message: 'mode: cloud', 'JOB ID: 5076866b52754', 'mode = cloud', 'eucarc = default', 'eucaImageEmi = emi-A8F63C29', 'instanceType = m1.small', 'pubkey = default', 'numOfNodes = 4', and 'applicationType = TwisterKmeans'. A button labeled 'click here resubmit a new job' is visible next to the 'List Jobs' menu item. The footer shows the Indiana University Bloomington logo.

Figure 7 b). Message after a job submitted

SASA HPC

Home

List Jobs

Job Status

About

SalsadPI Job Status Page

Job ID	App. Type	# of computer nodes	Job submission time	Job finish time	Job output	Status
5075e2d32dcae	HadoopWordCount	2	10/10/2012 05:04:19 PM EDT	10/10/2012 05:07:20 PM EDT	link	Finished
5075e2d7b7486	TwisterWordCount	2	10/10/2012 05:04:23 PM EDT	10/10/2012 05:07:51 PM EDT	link	Finished
5075e2e35026c	HadoopKmeans	2	10/10/2012 05:04:35 PM EDT	10/10/2012 05:09:40 PM EDT	link	Finished
5075e2e8ba43f	TwisterKmeans	8	10/10/2012 05:04:40 PM EDT			Running
5075e7c0b465e	HadoopWordCount	2	10/10/2012 05:25:20 PM EDT	10/10/2012 05:28:28 PM EDT	link	Finished
5076866b52754	TwisterKmeans	4	10/11/2012 04:42:19 AM EDT			Running


 Indiana University Bloomington

Figure 7 c). Job status page

SASA HPC

Home

List Jobs

Job Status

About

Enter Job ID:

Job 5076866b52754 is still running, please check again later

Program output is shown as following:

```

1349944939
mode = cloud
eucarc = default
eucaImageEmi = emi-A8F63C29
instanceType = m1.small
pubkey = default
numOfNodes = 4
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): 4
[INFO] Cloud mode start VM(s) with using chef
4 VM(s) are starting.....
.....
.....
.....

```


 Indiana University Bloomington

Figure 7 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 , 125.06291166821107 , 248.2027598988204 ,
Total Time for kmeans : 5.325
Total loop count : 11
4811 [main] INFO cgl.imr.client.TwisterDriver - MapReduce computation terminated gracefully.
-----
Kmeans clustering took 5.38 seconds.
-----
4829 [Thread-0] DEBUG cgl.imr.client.ShutdownHook - Shutting down completed.

```

Figure 7 e). Example of Twister Kmeans output

5. Show case: salsaDPI in Science Cloud Summer School 2012

In summer 2012, Indiana University and other 9 sites of university (<http://www.vscse.org/summerschool/2012/scss.html>), has held an virtual conference across nations. This conference mainly introduces and educates cloud technologies to graduate-level students and staffs. salsaDPI, online tutorial page is located at <http://salsahpc.indiana.edu/ScienceCloud/reproduce-intro.html>, has been taking part as one of the main tutorials of this activity. Figure 8 is one of the photos taken from this activity. It has shown the abilities of deploying single node and virtual runtime environment for Hadoop and Twister applications from raw OS environment (without any software pre-installed). Also, it automatically executes Hadoop and Twister applications such as WordCount and Kmeans once after installing the selected software stack (see Figure 3 for configuration details). Figure 9 shows the 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=kWom0lj8qxl (please view it with video quality of 1080p).



Figure 8. Science Cloud Summer School 2012

```

File Edit Format View Help
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# cat lsalsadPI_output/

```

Figure 9. running salsadPI HadoopWordCount on FutureGrid

6. Conclusion and Future Works

We have implemented a command-line runnable, programmable, and user-friendly online framework salsadPI for reproducible applications in dynamic computing environments. salsadPI still has room for improvement, especially, we have not reached the real research level yet. Possibilities such as performance analysis among related works, suitable use cases for scientific applications, and understanding of the internal data communications. So, in sum, research and software engineering possibilities are listing below

1. Test different runtimes, different algorithms, and various applications to understand their behaviors.
2. Record the internal communication time of each component of salsadPI.
3. Record and investigate SSH message redirect time and weight of salsadPI
4. Test salsadPI with different permanent storage such NFS, SFTP/HTTP, Object-storage Walrus.
5. How many nodes could salsadPI handle? What are the system behaviors when running salsadPI with multiple clouds?

Acknowledgment

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

Reference

- [1] *FutureGrid*. Available: <https://portal.futuregrid.org/>
- [2] J. Klinginsmith, M. Mahoui, and Y. M. Wu, "Towards Reproducible eScience in the Cloud," presented at the Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011.
- [3] J. Bresnahan, T. Freeman, D. LaBissoniere, and K. Keahey, "Managing appliance launches in infrastructure clouds," presented at the Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery, Salt Lake City, Utah, 2011.
- [4] *Opscode Chef*. Available: <http://www.opscode.com/chef/>
- [5] J. Fischer, R. Majumdar, and S. Esmaeilsabzali, "Engage: a deployment management system," *SIGPLAN Not.*, vol. 47, pp. 263-274, 2012.
- [6] K. Keahey, P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau, "Infrastructure Outsourcing in Multi-Cloud Environment," presented at the the 8th Open Cirrus Summit, San Jose, CA, 2012.
- [7] (2012). *Science Cloud Summer School 2012*. Available: <https://portal.futuregrid.org/projects/241>
- [8] *Oracle VM VirtualBox*. Available: <https://www.virtualbox.org/>
- [9] *JSON metadata format*. Available: <http://www.json.org/>
- [10] Open Stack. (2010, November 7). *Open source, Open standards Cloud*. Available: <http://openstack.org/index.php>
- [11] *Nimbus Cloud Computing for Science*. Available: <http://www.nimbusproject.org/>