Fall 2011

EC2LAB: SAAS USING AMAZON ELASTIC CLOUD COMPUTE

Manisha Gaikwad
San Jose State University

EC2LAB:

SAAS USING AMAZON ELASTIC CLOUD COMPUTE

A Master's Project

Presented to

Dr. Sami Khuri

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Manisha Gaikwad

December 2011

The Designated Thesis Committee Approves the Thesis Titled

EC2LAB:

SAAS USING AMAZON ELASTIC CLOUD COMPUTE

by

Manisha Gaikwad

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2011

Dr. Sami Khuri,	Department of Computer Science	Date
Dr. Jon Pearce,	Department of Computer Science	Date
Sachin Gaikwad	Software Engineer (Riverbed Technology)	Date

ABSTRACT

EC2LAB:

SAAS USING AMAZON ELASTIC CLOUD COMPUTE

By Manisha Gaikwad

The cloud computing is gaining popularity as it provides an infinite pool of hardware and software resources on demand. The Infrastructure-as-a-Service (IaaS) layer provides the physical resources, and relieves the users from the tedious as well as time consuming task of procuring and setting the server as well as the storage. This project harnesses the capability of the Amazon IaaS layer. The Software-as-a-Service (SaaS) application which is built on top of Amazon IaaS layer, helps the users to easily handle and connect with Amazon's Elastic Cloud Compute (EC2) instances.

Table of Contents

List of figures

1 Introduction

The cloud computing provides an effective solution to variable physical demands of any application. The ability to quickly procure, setup, and scale physical resources is an important feature provided by Amazon's Cloud via the Infrastructure-as-a-Service (IaaS) layer. This project uses the IaaS layer via a Software-as-a-Service (SaaS) web application to create a virtual lab environment for users. It helps users to easily create, connect, and terminate instances, thus providing a quick facility to handle multiple instances. It uses a flexible architecture to easily scale the resource consumption depending on the usage.

The Amazon Web Service (AWS) APIs are used to connect with IaaS layer of cloud and gain compute power on demand on pay-per-use basis. The compute power is provided by the Elastic Cloud Compute (EC2) product offered by Amazon Cloud. It provides several machine images to generate virtual computational instances ranging from micro (1 virtual core, 613 MB memory) to High CPU (8 virtual cores with 2.5 EC2 Compute Units each), and supports variety of operating systems (windows, Linux, etc).

This project “EC2Lab” uses the Amazon physical resources to instantiate 3 types of Linux machine images:

1. Small (1 EC2 Compute Unit - 1 virtual core and 1 EC2 Compute Unit)
2. Medium (4 EC2 Compute Units - 2 virtual cores and 2 EC2 Compute Units each)
3. Large (20 EC2 Compute Units - 8 virtual cores and 2.5 EC2 Compute Units each)

1.1 Objective

This project creates a lab environment for users (especially students) to work with a range of multiple core machines. Amazon Machine Images are instantiated on-demand using this Software-as-a-Service (SaaS) web application, which enables users to easily start, monitor, and terminate the instances via an intuitive web interface. Thus, the users can start an instance, work with several high performance multiple core machines, and release the instance when no longer required. It provides an excellent learning platform to explore physical resource serving capabilities of the cloud and helps to easily use the high performance physical resource from the cloud on on-demand basis.

This project creates a web application “EC2Lab” to control physical resources extracted from Amazon cloud. It uses an easy, intuitive interface to generate and control the virtual instances. This web application is built using Amazon PHP SDK, and the Amazon Elastic Cloud Compute (EC2), and enables users to work with virtual instances without going through the process of Amazon account creation, key generation, certification generation, and Amazon Machine Image (AMI) creation. This project handles all the account setup, and AMI image generation in the back-end. It thus enables users to start running their applications on the Amazon infrastructure with application's log in credentials (EC2Lab user-name and password.)

This application aims to serve 2 types of users -

1. User – The users are students who will be provided with the username and password. They can start, connect, and stop instances. They can also view the status of currently running instances in the web application's dashboard.

2. Administrator – An administrator will be able to monitor all instances started by users. He can terminate the instances that are not required.

1.2 Research Scope

The research scope includes:

1. Understanding cloud computing technology using Amazon Web Services
2. Comparison with other cloud providers
3. Understanding the various machine images and instantiation process
4. Exploring the Amazon PHP SDK to create the application
5. Understanding the Configuration Setting for Amazon Web Services to create instances

1.2.1 Cloud Computing using Amazon Web Services

Besides Amazon, there are several cloud providers available such as Google App Engine, and Microsoft Azure that host a web application. This project evaluates the Amazon cloud infrastructure in terms of price, stability, and services. It explores the Amazon Web Services Layer which provides the lower level infrastructure service for better control over the deployed application. The cloud concepts – layers, deployment model, features, Amazon EC2, and S3, are explored in-depth.

1.2.2 Comparison with other cloud providers

The Amazon Cloud is compared with other cloud providers like Eucalyptus to find the ease of use, functionality differences, and to determine the best IaaS platform to base this project upon.

1.2.3 Understanding the various machine images

There are several EC2 instances available such as: Small, medium, and large. Some instances are used for high compute power and others for high memory capacity. This project focuses on compute power of the machine instances. All the instances with their pricing are studied to select the machine images suitable for this project.

1.2.4 Exploring the Amazon PHP SDK to create the application

The web application is built on top of Amazon EC2. The PHP SDK provided by Amazon is explored to determine the feasibility of the project. The PHP SDK supports the web application and facilitates the use of Amazon Web APIs for this project.

1.2.5 Understanding the Configuration Settings for Amazon Web Services

The pre-requisites such as feasibility parameters, bundling, and configurations of various compute instances on Amazon cloud platform are studied. The setup and configuration serves as building blocks for this SaaS application.

In the next chapter, the cloud concepts such as – layers of cloud, deployment models, technology adoption curve, and Amazon IaaS layer is discussed.

2 Cloud Concepts

The cloud concept was introduced by IBM in 1960 to lease resources on demand, but due to lack of supporting infrastructure like high speed Internet, high computational power, and advance web applications, this vision couldn't be realized. The era of cloud computing has emerged again as it ranks at the top position in the "10 CIO technology priorities for 2011"[15].

This technology enables provisioning of hardware and software resources on demand over the Internet. The cloud concept is similar to renting a resource using a pay-per-usage charging structure. Users can rent a hardware resource, platform, or complete software application over the Internet. The key features include:

1. Easy scale up and scale down facility
2. Easy monitoring of usage
3. Security provided by the cloud provider

The next sections explore cloud concept in detail and provide information about the deployment models, cloud layers and adoption curve. The deployment model contains public cloud, private cloud, hybrid cloud, and community cloud. The cloud layers include Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The technology adoption cycle section covers discussion about the current position of the cloud technology according to the market research.

2.1 Deployment Models

There are four basic deployment models for cloud: Public, Private, Hybrid, and Community [22][24][25]. These are described as follows:

1. *Public Cloud:* The general user or organization can host their application on public cloud. It's a shared environment for all users. The users are charged on the basis of the resource usage. The public cloud providers are Amazon, Google App Engine, and Microsoft Azure among many others.

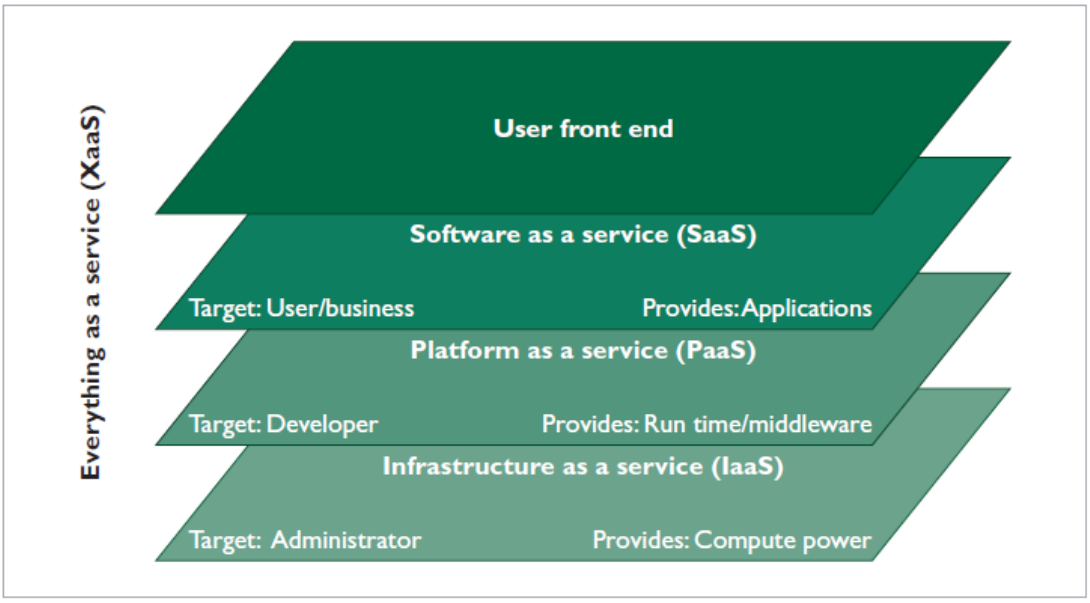
2. *Private Cloud:* The private cloud is not available to general public. It is restricted to an organization or a group. Usually they are internal to an organization. They provide better security as critical data access is limited to the organization. VMware or Citrix are the examples of the private cloud providers.

3. *Community Cloud:* When several organizations or communities want to use an infrastructure for resource sharing and common concerns, the community cloud provides a better solution. The IlliniCloud[17] created by Illinois University is an example of community cloud.

4. *Hybrid Cloud:* Hybrid cloud is a combination of two or more deployment models. For example, this cloud may contain both: the private and public cloud. The private cloud can host the application with critical data; whereas, the public cloud can hold the non-critical data applications.

2.2 Layers of the Cloud

Figure 1: Layers of Cloud Infrastructure [27]



According to the Figure 1, there are three layers of cloud and the highest layer is SaaS. These layers provide different services to the users. The next subsection provides details about each cloud layer:

1. *Infrastructure as a Service (IaaS):* It is the bottom layer of the cloud framework. It provides physical resources such as computation power and storage to an application. The Amazon Web Services provide infrastructure resources – EC2 provides the virtual instance for computation and S3 provides the storage for Internet applications. This project uses the infrastructure services provided by Amazon to host a web application.

2. *Platform as a Service (PaaS):* The PaaS is the middle layer which supports application developers to perform the development and carry out the development life cycle. It provides the development environment without installing software on the physical machine. It is a middleware for providing runtime environment without any software installation. Force.com platform service is an example of PaaS.

3. *Software as a Service (SaaS):* The complete software application can be used by the clients. It is the top most layer of the cloud framework. The application is delivered to the end users over the Internet. Base Camp, Quicken online, Gmail, SalesForce (CRM application) are examples of SaaS.

2.3 Technology Adoption Cycle of the Cloud

This section is used to analyze the market to denote the position of the cloud computing on the technology adoption curve [24].

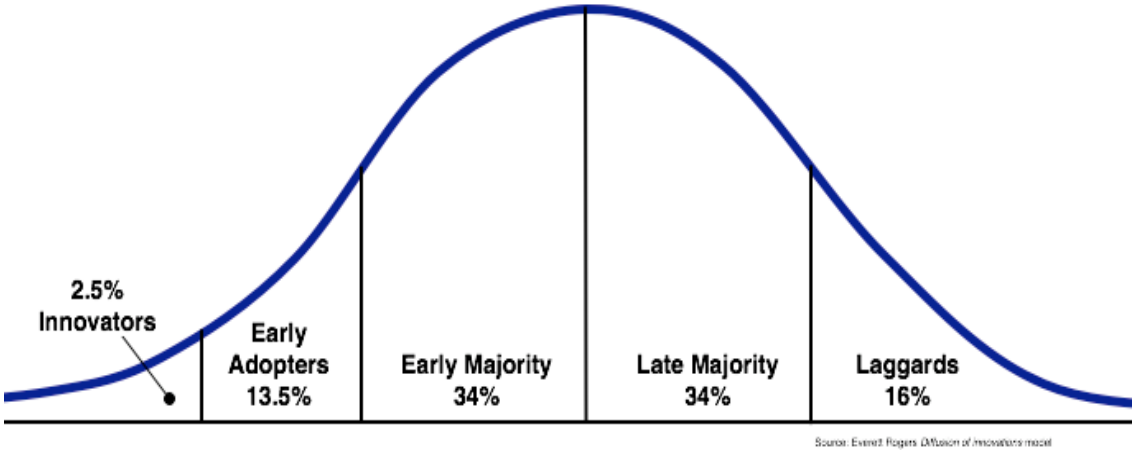


Figure 2: Technology Adoption Curve [24]

The technology adoption curve displays the phases and percentage distribution of the users in each phase. This curve can be applied in a generalized manner to any new technology. The first phase of the curve contains the innovators, they adopt the technology to explore the functionality, they do not focus on the purpose of the technology and their sole focus is the technology itself. Next comes the early adopters category, which relate to the purpose of the technology and use it if the technology fulfills their purpose. Later, the early majority performs thorough analysis and adopts it when the technology strongly meets their demands. The second last phase is late majority, they are not technically sound and skeptical to change their conventional usage. The last people to adopt a technology are called laggards; they use the technology when it is deeply integrated in some other product. The laggards are not aware of the existence of the technology. Figure 2 shows the technology curve with different phases and the cloud platform's position lies in the early adopters phase. This position is based on the market analysis and revenue generated by Amazon, Google, and Microsoft in 2010-2011[8].

2.4 Amazon IaaS Services

The Amazon's Elastic Cloud Compute(EC2) and Simple Storage Service (S3) are used for this project [2][3] which lies in the IaaS layer. The EC2 and S3 are accessed using Amazon Web Service APIs. The EC2 provides capability to create virtual instances on the public cloud. These instances can be remotely customized to deploy customer’s

application. The S3 allows the users to store data in the cloud using the region specific buckets. Figure 3 displays the AWS architecture with EC2 and S3.

The following are significant features of Amazon Web Service (AWS):

- The users are charged according to the usage. The users can easily monitor their usage using Amazon Console activity and usage reports.
- The users can procure infinite resources. They can easily acquire new resources or scale down as per their requirement.
- The users can choose any operating system for instances. The AWS supports Windows, Linux, Fedora, and Ubuntu operating systems.
- It provides optimization and easy migration facilities.

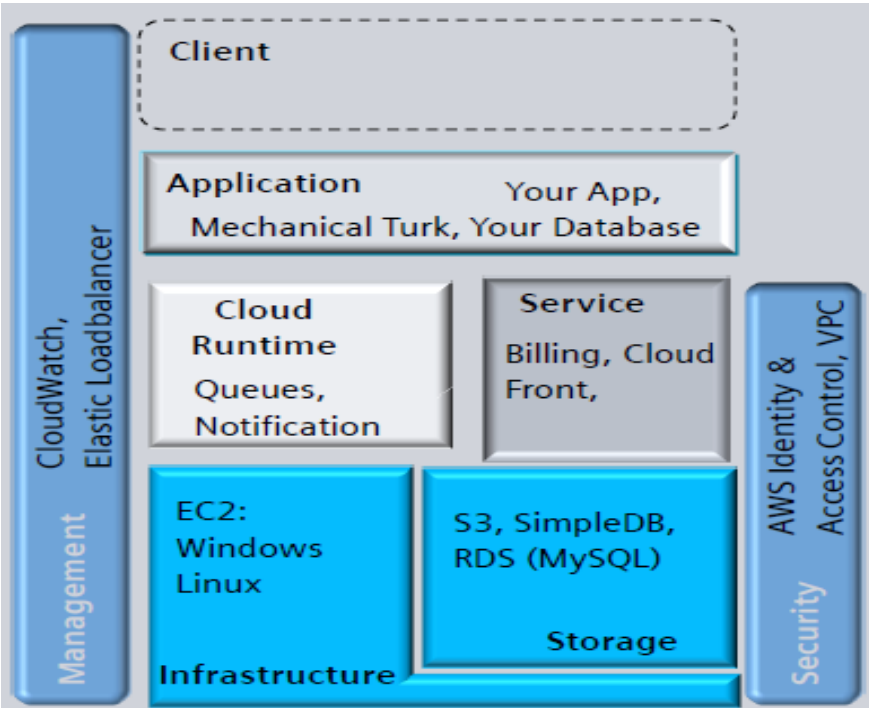


Figure 3: Amazon Web Service Platform [3]

2.4.1 Elastic Cloud Compute

The Elastic Cloud Compute is used to provide computational power to the applications. The EC2 creates instances on the public cloud. These created instances are virtual machines which provide CPU resources to the applications, enabling them to run over the Internet. The users are relieved from the physical resources procurement and setup task, which might take weeks to deploy an application. The instance can be easily scaled up or scaled down to meet the computational demand. The EC2 provides different types of instances to suit any resource requirement of the users. The standard instances frequently used are described in the following subsections [2][3]:

1. Small Instance: It provides memory of size 1.7 GB, CPU Compute Unit 1 , 32-bit OS, moderate Input/Output, and instance name is m1.small.

2. Large instance: It provides memory of size 7.5 GB, CPU computer units 4, 64-bit operating system, high Input/Output, and instance name is m1.large.

3. Extra Large Instance: It provides memory size of 15 GB, CPU Compute Units 8, 64-bit operating system, high Input/Output Performance, and instance name is m1.xlarge.

4. Micro Instance: It provides memory size of 613 MB, maximum 2 CPU Compute Unit, 32/64-bit operating system, low Input/Output performance, and instance name is t1.micro.

2.4.2 Simple Storage Service (S3)

Simple Storage Service allows users to store data over the Internet using a monthly payment plan. There are two modes of storage [2]: standard storage and reduced redundancy. The standard storage makes several copies to provide 99.999999999% of durability. In reduced redundancy mode, limited copies are made and it provides 99.99% data durability. The S3 uses the buckets and objects to organize the user data. The bucket is a top layer of the organization and next layer is the object. The bucket provides structure for storage, similar to the folder in operating system. The buckets are created for a specific region and name must be unique. Several operations such as add, delete, view and move can be performed on them. The Figure 4 displays the sequence of steps to create a bucket. The following steps are performed to store data using S3:

1. Create an Amazon Web Service account and sign up for S3 service. The user is assigned access key and private key. This access key can be used with application management console or with bucket explorer application on your own machine.
2. Log in into Application management console and create a bucket. The buckets will be created specific to a region. The region which is closest to the user can be selected for faster access of data. The data in S3 bucket can also be migrated using AWS migration service.
3. Upload the data file or folder from the local machine into the bucket, this data is called the object.
4. Several operations can be performed on the object. The buckets can also be deleted if not used.

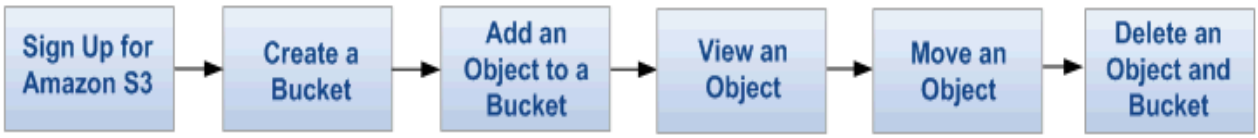


Figure 4: S3 Operation Cycle [2]

2.5 Eucalyptus Cloud

The Eucalyptus cloud is similar to Amazon Cloud [13]. It is an open source cloud platform that provides APIs compatible with Amazon APIs. The Eucalyptus cloud provides infrastructure for creating the private cloud. This service was deployed on Ubuntu system for the feasibility study of Eucalyptus platform. The process revealed several disadvantages compared to Amazon:

1. Installation problem: The software was difficult to install and required several patches to the kernel for installation.
2. Support for different OS: It supports very few operating systems. The windows OS is not supported for installation or for image instantiation.

Due to the above limitations, the Eucalyptus open source cloud was not used for this project.

In the next chapter, the configuration settings needed for using the EC2Lab web application are discussed.

3 Configuration

3.1 Configuring and Bundling Amazon Instances

This section provides the information regarding configuration settings for the Amazon account, and pre-requisites needed to customize a Amazon machine images using Amazon Web Services (AWS)[2][28]. An Amazon Web Service account is setup with active EC2 and S3 accounts. Later, a machine image is bundled to save the changes that are made to the running instances.

To use EC2 and S3 for this project, the following steps are performed [3]:

1. Create the S3 and EC2 accounts
2. Create the image
3. Modify the image with remote connection to virtual instance
4. Delete a running instance and clean up

3.1.1 Creating Amazon Accounts

The S3 and EC2 accounts can be created using Amazon Web Services web site[2]. The users will receive their access credentials, sign-up credentials and the account identifier. The access credentials include the public access key, secret access key, and X.509 certificate with associated private key. The sign-up credentials are user-name and password. The account identifier is a unique number associated with the account. The

X.509 certificate has to be regenerated in case the private key is lost, since private key is not stored on Amazon web site. The credit card details are essential to open both the accounts. The users can easily monitor their activity report and the usage charges using the Amazon Management Console. Figure 5 displays the Amazon Management Console Dashboard. Users can browse the S3, EC2, and other services using this dashboard. The dashboard displays the summary about running instances, key pairs and security groups. New instances can be launched using the launch button. The Java Runtime Environment (JRE) is a pre-requisite for running the cloud application; therefore, it should be installed on the local machine which serves as a client for the web application.

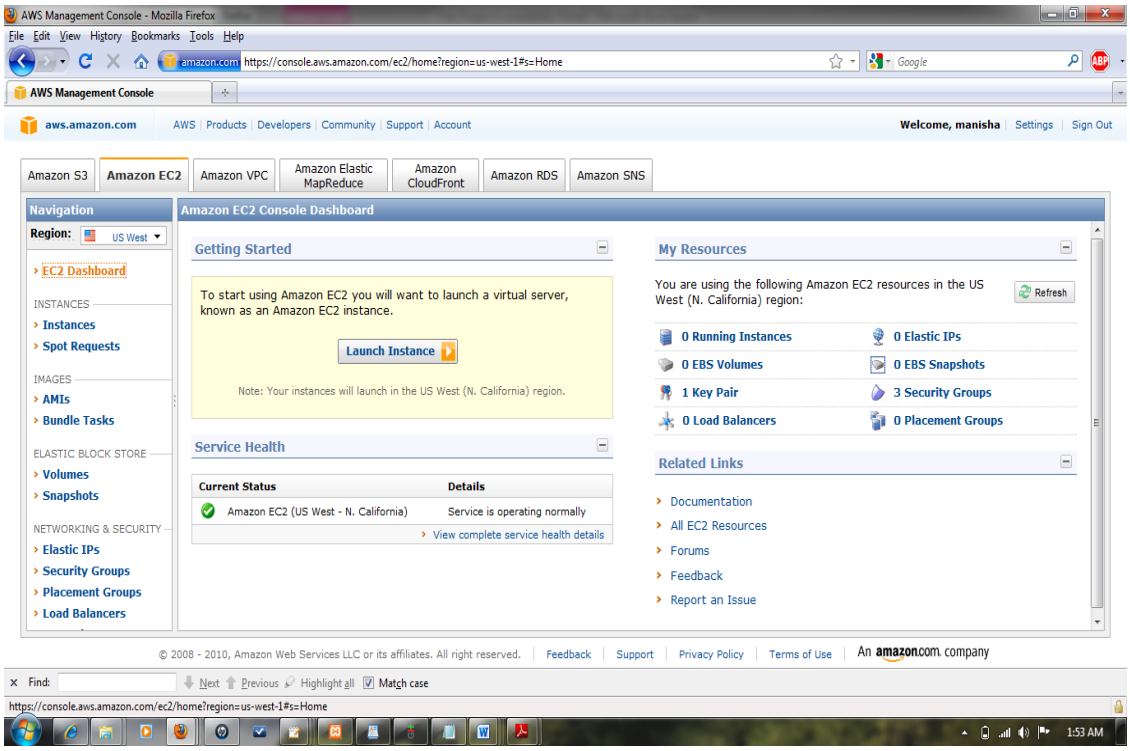


Figure 5: Amazon Management Console Dashboard Screen Shot

Steps are listed to create and setup amazon account as follows:

1. Goto Amazon Web Services: http://aws.amazon.com/ → Create an AWS Account
2. Provide Log in Credentials
3. Provide contact information
4. Provide credit card details
5. Perform phone authentication and verification
6. Account will be activated
7. Activate the product account
    a. S3 account activation and
    b. Amazon Ec2 account
8. Save Security Credentials key pairs and account keys
9. Check Usage Reports if required

After the account is created there are three methods to interact with the Amazon services – shell commands, Elastic Fox, or Amazon Management Console:

Shell: The user has to obtain the command line tools from the Amazon Resource Center to use the shell. The user has to provide the X.509 security certificate and private key.

Elastic Fox: It is a Firefox browser add-on which can be easily installed to perform EC2 and S3 management. The Amazon account has to be configured to access elastic fox.

Amazon Management Console: It is an online application management console of AWS. Among all the options explored, the simplest to use is AWS management console.

3.1.2 Instantiating the Machine Image

The machine image is a package with operating system, files, software tools and libraries needed to run an application[3]. This image is used to generate the virtual machine instance on the cloud. The users can use API ec2-describe-image to check several public images using the shell prompt. The users can launch an instance from the machine image. The instance can be launched from an existing image or the users may chose to upload a local machine image for instance generation. It is convenient to modify an existing image with user files and libraries to run a custom application. There are several ways to connect to an instantiated image this project uses the Linux shell terminal and the SSH command for connecting to an existing virtual instance.

The Figure 6 shows the process of launching an instance and connecting to Linux or windows system for modification. The connection process for both operating systems is explained in the later section.

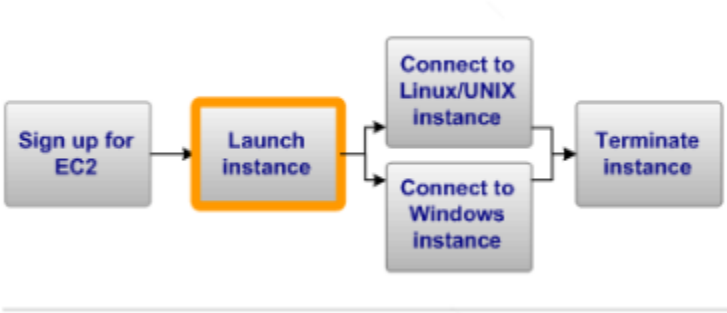


Figure 6: Launch an Instance [3]

3.1.3 Modify the Image with Remote Connection to Virtual Instance

PuTTY, SSH, and Remote Desktop Connection (RDC) tools are helpful for modifying an existing image, by setting up remote access to the running instance. SSH is used in this project for connection and modification of the virtual instance. The running instance can modified after connection is established successfully. The image has to be bundled again in an S3 bucket, whenever a running instance is changed.

PuTTY (Linux instance connection): First step is to convert the private key .pem file to .ppk using PuTTYgen. Next step is connecting using PuTTY SSH(See Figure 7).

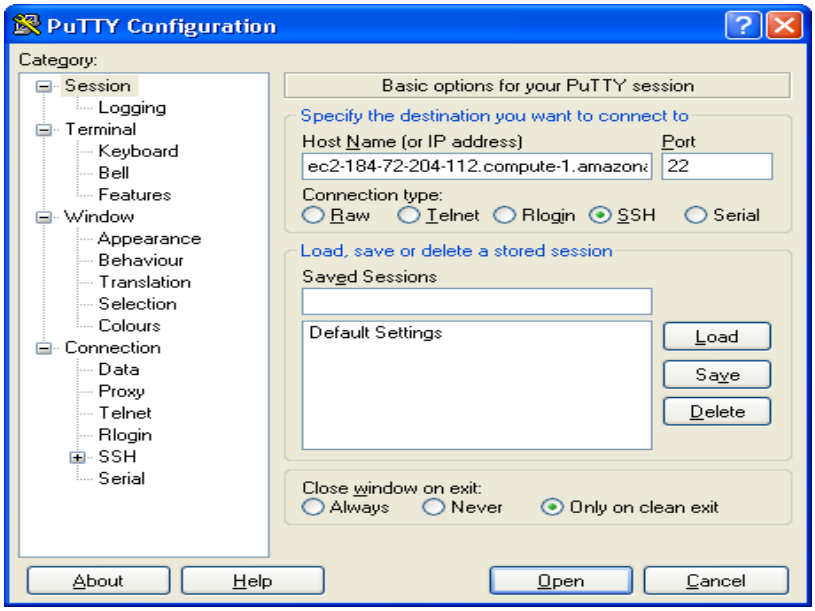


Figure 7: Putty Configuration Window [3]

SSH: SSH command can be used from Linux command prompt to connect to the generated instance. The SSH uses the key and DNS parameter for establishing the connection.

Example: ssh -i keypair1.pem ec2-user@ec2-50-17-170-92.compute-1.amazonaws.com. The first parameter is the path of the key and second parameter is the DNS name of the server where connection has to be established.

Remote Desktop Connection (RDP): The Windows Administrator password is retrieved from the running connection to establish the remote connection. The private key is decrypted using the retrieved administrator password (See Figure 8). Finally, the remote connection is established using RDC application.

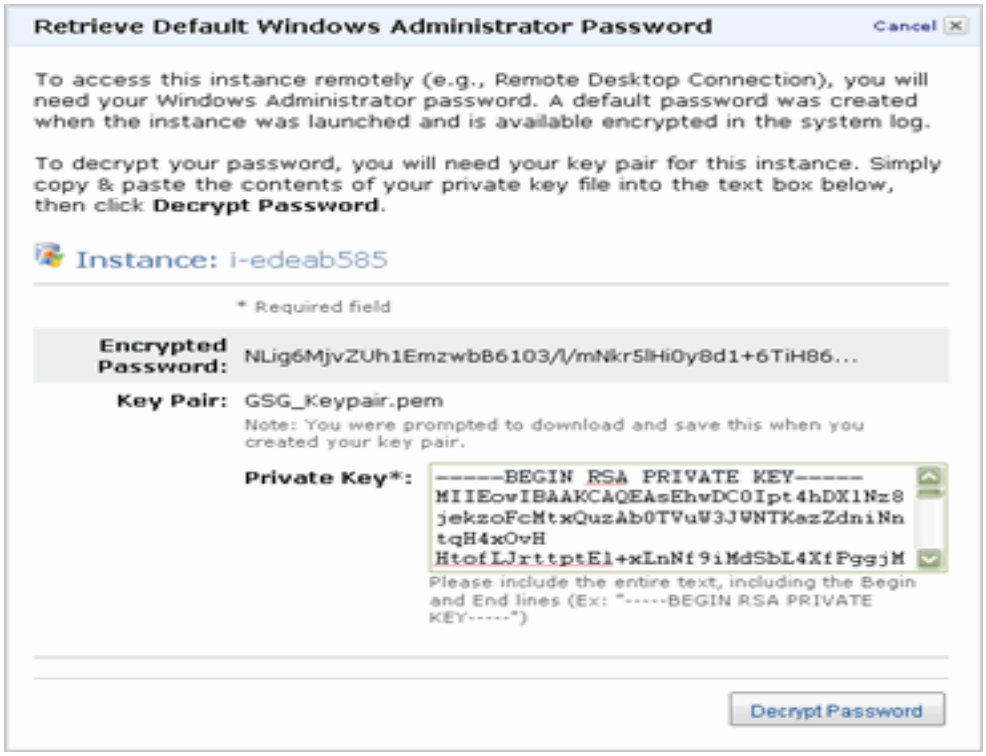


Figure 8: Decrypting a Private Key [3]

3.1.4 Bundling the instance

The steps described in this section provide bundling information for amazon instances. The bundling process is necessary to save the public image as individual private image after changes are made to the running public instances. The bundled updated image can be saved in S3 bucket. Later, it should be registered as AMI to instantiate it [3][28].

Step to bundle AMI using Amazon AMI tools:

1. Copy the the account credentials i.e private key and certificate to instance using the scp shell command:

> (sudo) scp -i keypair1.pem private_key.pem certificate ec2-user@ec2-50-17-170-92.compute-1.amazonaws.com:/home/ec2-user

2. Connect to the instance using ssh:

> ssh -i keypair1.pem ec2-user@ec2-50-17-170-92.compute-1.amazonaws.com

3. Log in to the instance after connection and copy keys to mount from home:

> copy the keys to mnt as ec2-user: cp /home/ec2-user.*.pem /mnt

4. AMI tools should be downloaded on virtual image to bundle the running, by using the the steps to obtain the AMI tools on the instance:

> # curl http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools.zip > ec2-ami-tools.zip
>
> # mkdir ec2

cp ec2-ami-tools.zip ec2

cd ec2

unzip ec2-ami-tools.zip

ln -s ec2-ami-tools-* current

vi ~/.bashrc

5. Add the following lines at the end of the bash:

export EC2_AMITOOL_HOME=~/ec2/current

export PATH=${PATH}:~/ec2/current/bin

And then re-initialize the configuration file

source ~/.bashrc

6. Go to home:

sudo bash

7. Bundle the running instance using the AMI tools:

ec2-bundle-vol -d /mnt -k /mn/private_key.pem -c /mnt/certificate.pem -u XXXXXXXXXXXX

uid without - (from web account credentials)

8. Upload the bundled image to s3 bucket

ec2-upload-bundle -b linux_image_bucket -m /mnt/image.manifest.xml -a access_key -s secret_key

9. Log in web UI & registered new image

path : linux_image_bucket/image.manifest.xml

10. Launch & connect

ssh -i /home/hadoop/project-297/key/keypair1.pem ec2-user@ec2-50-17-170-92.compute-1.amazonaws.com

3.1.5 Terminate/Delete a Running Instance and Clean Up

The running instances are charged for the usage; therefore, any unnecessary running instance should be stopped. The ‘terminate’ option is used to stop the running instances. It is also possible to de-register an instance and delete your machine image from the S3 bucket. The de-registering and image removal task can be performed using graphical user interface or by using shell API ec2-deregister and ec2-delete bundle respectively.

In the next chapter, the application details such as - software requirements, project architecture, pricing details, and users of the application are discussed.

4 Application Details

4.1 Software Requirements

This section gives details about the software requirements for setting up “EC2Lab” SaaS application environment.

Software Requirement:

1. PHP
2. HTML
3. MySQL
4. Apache
5. PHP SDK

The Lamp stack is used in this application, as it provides PHP, MySql, and Apache. The Lamp stack is installed on the Ubuntu system for this project.

The PHP SDK[4] is provided by the Amazon to support web application development using Amazon APIs. The APIs used for this project are:

1. run_instances: This API is used to start the instance.
2. terminate_instances: This API will terminate the instance.
3. describe_instances: This API is used to provide the details about the instance.

4.2 Project Architecture

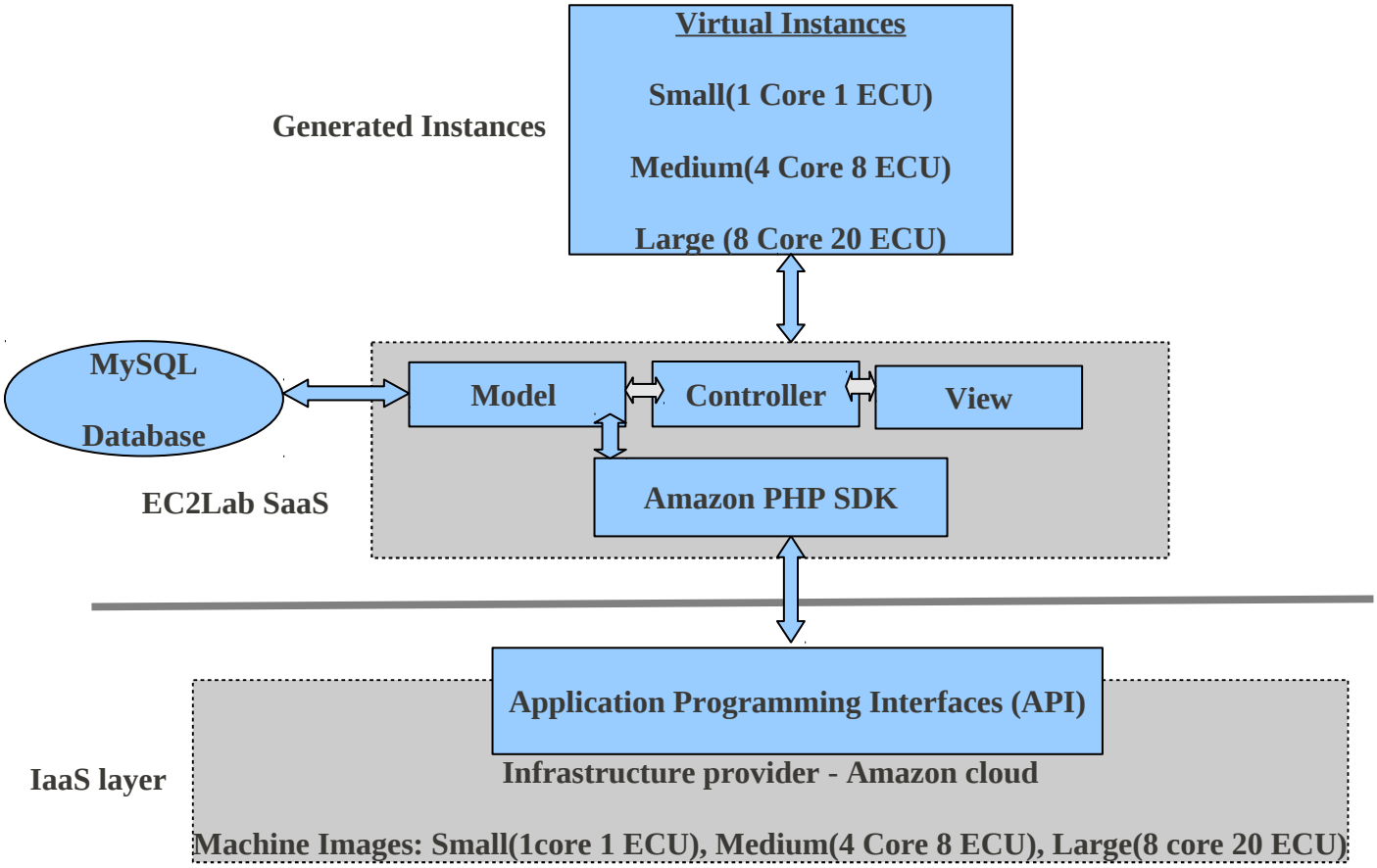


Fig 9: EC2Lab Project Architecture

There are 2 main layers of this web application - Iaas Layer and EC2Lab SaaS layer, as can be seen in the Figure 9.

IaaS Layer: This project uses the infrastructure layer of Amazon cloud to procure the physical resources. There are 3 types of virtual machines that can be instantiated- small, medium,and large to full-fill the physical machine requirement. The small machine image contains 1 core and 1 Elastic Compute Unit, Medium has 4 core 8 Elastic Compute Unit),

and large contains 8 core 20 Elastic Compute Units. The application programming interface(APIs) are used to connect the amazon IaaS layer to the SaaS layer.

SaaS layer: The EC2Lab SaaS application is built on top of the Amazon IaaS layer. It uses the APIs provided by the Amazon PHP SDK to communicate with IaaS layer to procure or terminate the virtual instances generated from the machine images. The PHP SDK is part of the SaaS application and communicates to IaaS layer using APIs. This application is built using Model-View-Controller design pattern. It provides an intuitive web interface to users and administrator to handle virtual instances. The details about model-view-controller pattern are provided in the next section.

Virtual Instances:

The SaaS application generates 3 types of virtual instances. Details about each virtual instance is provided in what follows:

1. Small –

   Processing power: 1 core with 1 Compute Unit

   Processing speed: 2.40 GHz

   Main memory: 1.7 GB memory

   Instance storage: 160 GB

   platform: 32-bit

   Name: m1.small

2. Medium –

Processing Power: 4 cores with 8 Compute Units

Processing speed: 2.40 GHz

Main memory: 15 GB

Instance storage: 1,690 GB

Platform: 64-bit

name: m1.xlarge

3. Large –

Processing Power: 8 cores with 20 Compute Units

Processing speed: 2.40 GHz

Main memory: 7 GB

Instance storage: 1,690 GB

Platform: 64-bit

name: c1.xlarge

4.3 Pricing Details

This project uses the Small, Medium and Large AMIs. The administrator is responsible to provide the credit card details and pay for the usage. The pricing details are as follows:

1. Small AMI (Linux, 1 core) - $0.085 per hour
2. Medium AMI(Linux, 4 Core) - $0.68 per hour
3. Large AMI(Linux, 8 Core) - $0.68 per hour

4.4 Model-View-Controller Architecture

This web application is based on Model-View-Controller(MVC) design pattern as described in the project architecture Figure 9. The MVC design pattern is used to divide the code into logical areas for better structuring, easy maintenance and reuse. The model handles the back-end connections the MySQL server and stores the business logic of the application. The view takes care of the presentation logic, and controller handles the communication between the model and the view.

Model classes:

1. AccountOperations.php – This class performs all the sign-in and account management for the user.
2. AjaxStart.php and AjaxStop.php – It performs the polling operation to provide real-time feedback when the instances starts and terminates.
3. DatabaseAccess.php – Contains basic calls used by all model classes to connect or disconnect mysql database.
4. InstanceOperation.php – Calls the Amazon PHP SDK APIs to instantiate or terminate the virtual instances.
5. UserAdminDataOperatios.php – Contains MySQL updates for user and admin accounts when instances are generated or terminated.

View:

The view contains the presentation logic of the application. The classes contained in this application are:

1. AdminView.php: Display pages for administrator

2. UserView.php: Displays user pages
3. Display.php: Displays home page
4. InprogressView.php: Display the realtime polling information to the amazon server, uses Ajax for update.

Controller: Server as event handler for the View. It sends the request to the model, and conveys the model's response back to the View.

1. AdminController.php: Handles the request and response for the administrator.
2. UserController.php: Handles the request and response for the user.
3. Signin.php: Handles signin operations for user and administrator.
4. DefaultController.php: Handles the home page requests

4.5 User for the Application

This application has 2 types of user -

1. **User**
2. **Administrator**

User: The user is provided with the user-name and password to access the application. The user can perform the following functions:

1. **Start the Instance** – Start the virtual machine from small, medium, large machine images.
2. **Terminate the Instance** – Stop the running instances started by him.

3. **Connect to the Instance** – Use the ssh command to connect to the instances started by him
4. **Check Summary of the Running Instances** – The user is provided with the dashboard where he can check the status of all the instances that are started by him. He can terminate one or several instances from the dashboard.

Administrator: The administrator can perform the monitoring operation of all the running instances. He has super user privileges to check and terminate the instances started by any user. The administrator can perform the following operations -

1. **Check Summary:** The administrator can check all the instances started by the users.
2. **Terminate:** The administrator can terminate instances started by other users.

4.6 Screen-shots of the Application

Home page

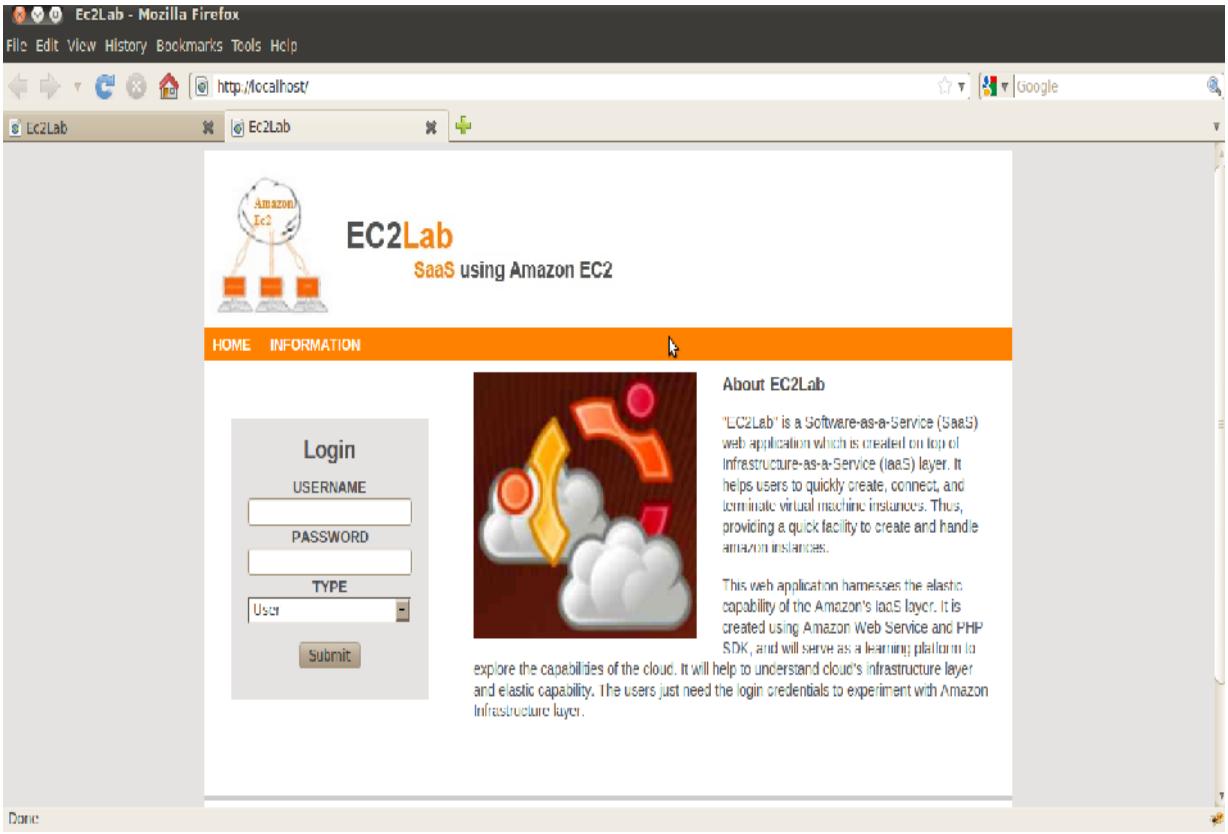


Figure 10: Screen-shot for home page

Figure 10 is the screen-shot for the home page of the EC2Lab application. The users (user or administrator) can log in to the application using the home page. It provides information about the EC2Lab application and contains some useful links related to the application.

User Dashboard page

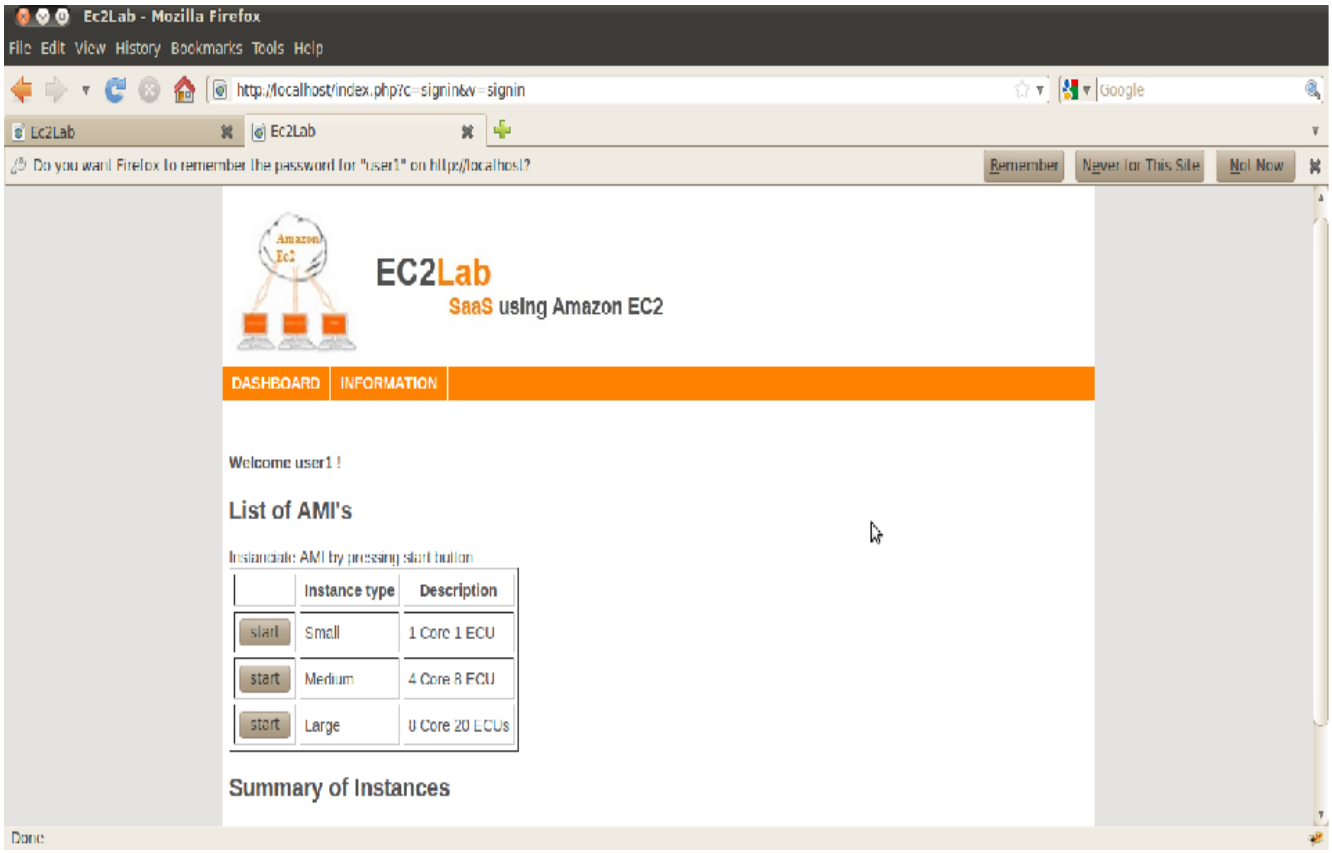


Figure 11: Screen-shot for the User dashboard

Figure 11 shows the user dashboard, it contains the list of AMIs that can be instantiated by the user, and summary of instances.

Admin DashBoard

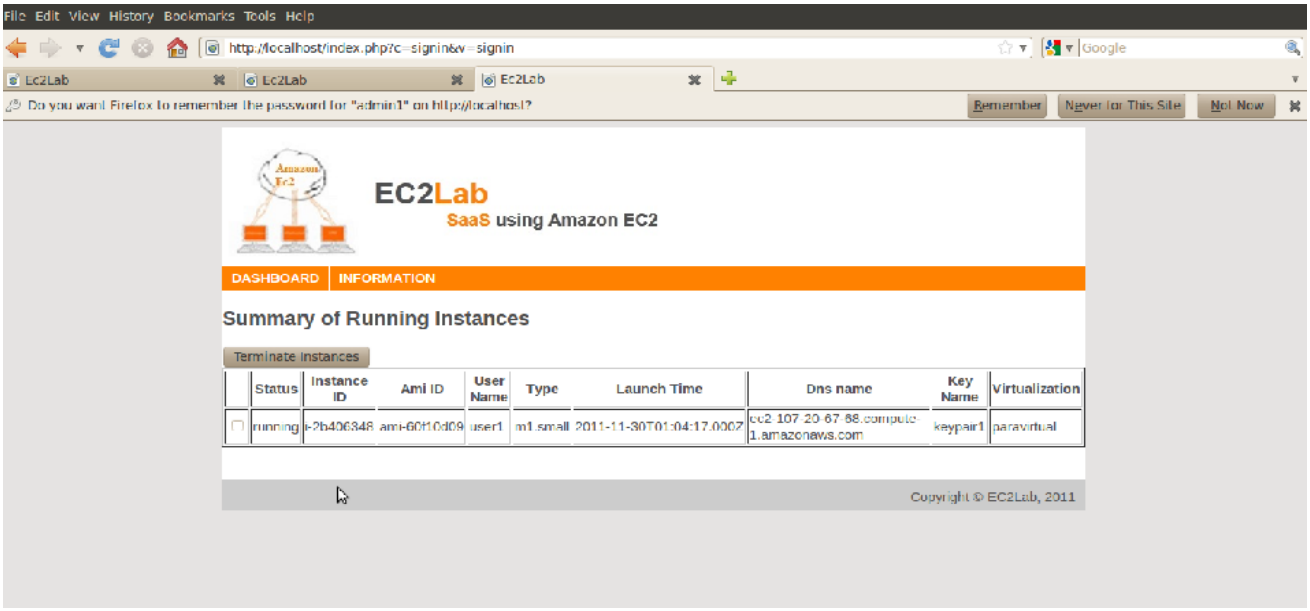


Figure 12: Screen-shot for Administrator Dashboard

Figure 12 shows the administrator dashboard, this page contains summary of running instances. The administrator can also terminate instances from this page.

In-progress View

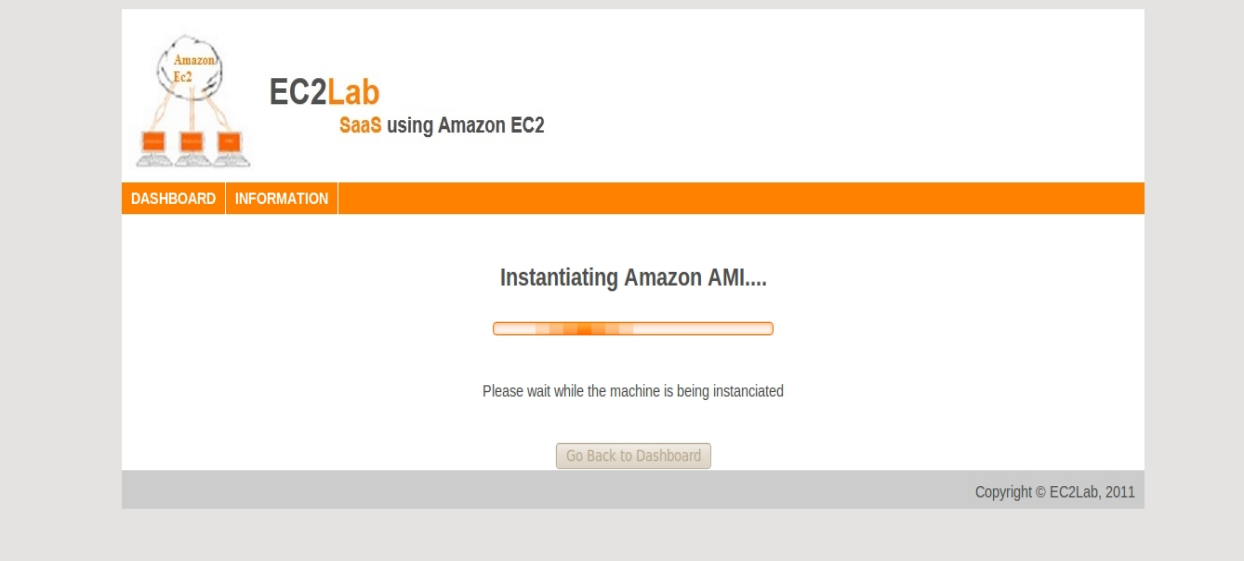


Figure 13: Screen-shot for In-progress View

Figure 13 screen-shot shows the in-progress view. This view is displayed when the Amazon image is being instantiated.

Instantiated Display Information

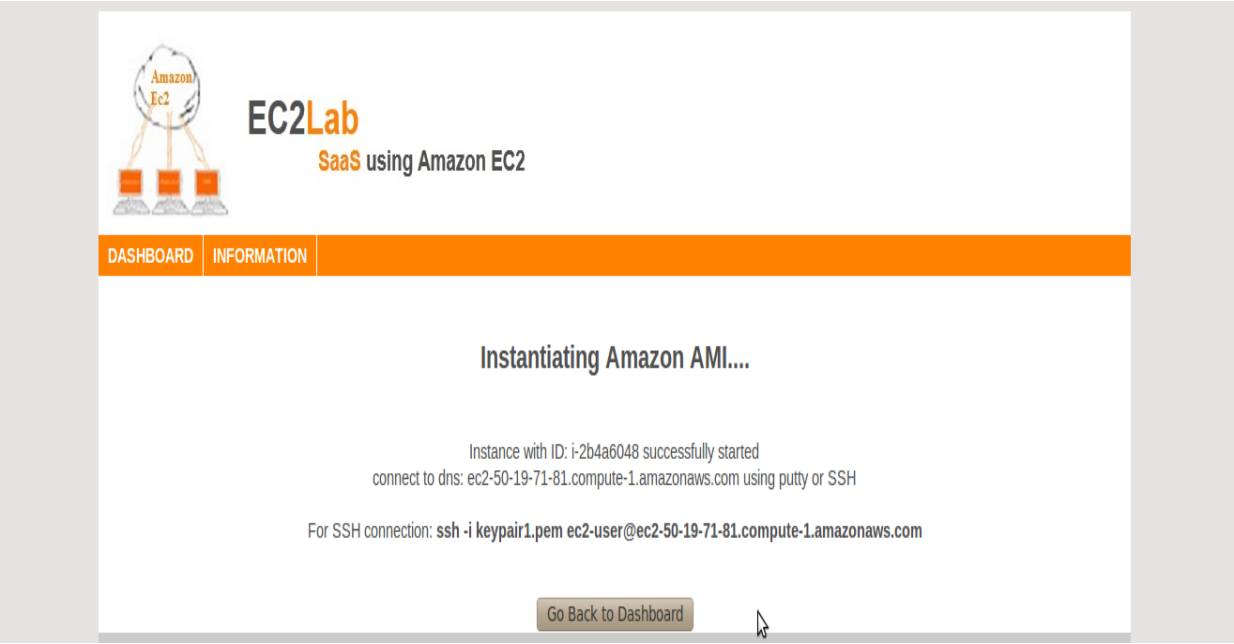


Figure 14: Screen-shot for instance status and connection information

Figure 14 displays the instance information after the Amazon machine image is instantiated.

The next chapter discusses the test results of physical machine and Amazon virtual instances. The analysis of the test result is also provided.

5 Test Results

An application written in C with OpenMP directives is used as sample test code. The program contains OpenMP directives for parallelism, and it is compiled using OpenMP capable gcc compiler [Appendix B]. Tests are carried on 1 core, 4 cores, and 8 cores machines with varying number of threads on -

1. Virtual machines generated using EC2Lab
2. Physical machines of similar configuration

The results are compared to check the performance offered by Amazon virtual machine against the physical machine with similar configuration.

The sample C code with OpenMP directive is given in Appendix B, it is based on project 2 of the CS286-Advanced Parallel Programming course by Dr. Robert Chun (Professor CS department San Jose State University). The program calculates the value of PI in 1000000000 steps using several threads. The work is distributed to several cores by varying the number of threads using omp_set_num_threads directive. The pragma directive from OpenMP is used to parallelize the for loop. The private variable and the result variable are also specified for thread synchronization.

5.1 Test Results for Amazon Virtual machines

The sample code specified in the Appendix B is executed on the virtual instances(small,medium, and large) generated using the EC2Lab SaaS application. Configuration details virtual instances can be obtained from Section 4.2 project architecture, but quick summary is provided below:

1. One Core:

    Processing power: 1 core with 1 Compute Unit, 2.40 GHz

    Main memory: 1.7 GB memory

2. 4 Core:

    Processing power: 4 core with 8 Compute Unit, 2.40 GHz

    Main memory: 15 GB memory

3. 8 Core:

    Processing power: 8 core with 20 Compute Unit, 2.40 GHz

    Main memory: 7 GB memory

Table 1 shows the execution time for the parallel sample code, when executed on 1 core, 4 core, and 8 core Amazon virtual instance.

No. of Cores	Sequential	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
1	24.21 Sec	24.29 Sec	24.31 Sec	25.01 Sec	24.37 Sec	24.29 Sec	24.35 Sec
4	25.58 Sec	13.30 Sec	6.63 Sec	6.69 Sec	6.65 Sec	6.66 Sec	6.65 Sec
8	24.29 Sec	12.46 Sec	6.23 Sec	3.35 Sec	3.44 Sec	3.37 Sec	3.40 Sec

Table 1: Test results for amazon virtual instances

The result table shows the time taken by the parallel program when executed on 1,4, and 8 cores with varying number of thread 1 to 64. For one core, the execution time is

constant i.e. around 24.30 seconds for all the threads. For 4 core the execution time is constant after 4 threads, the work cannot be distributed further. Similarly, 8 core shows constant execution time after 8 threads.

5.2 Test Results for Physical Machines

Configuration details for physical machines are as follows:

1. One Core:

   Processor: Intel(R) Xeon(R) CPU, E5420 @ 2.50GHz

   Main Memory: 8 GB

2. 4 Core:

   Processor: Intel(R) Xeon(R) CPU, E5420 @ 2.50GHz

   Main Memory: 15 GB

3. 8 Core:

   Processor: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz

   Main Memory: 16 GB

Table 2 shows the execution time for the parallel sample code, when execute on 1 core, 4 core, and 8 core physical machine.

No. of Cores	Sequential	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
1	20.21 Sec	20.29 Sec	20.31 Sec	21.01 Sec	20.37 Sec	20.29 Sec	20.35 Sec
4	21.24 Sec	10.62 Sec	5.32 Sec	5.41 Sec	5.40 Sec	5.35 Sec	5.45 Sec
8	20.21 Sec	10.15 Sec	5.33 Sec	2.61 Sec	2.64 Sec	2.69 Sec	2.60 Sec

Table 2: Test results for physical machines

Table 2 shows the time taken by the parallel program when executed on 1,4, and 8 cores with varying the number of thread 1 to 64. For one core the execution time is

constant i.e. around 20.21 seconds. For 4 core the execution time is constant after 4 threads as work cannot be distributed further. Similarly, for 8 core processors the execution time is constant after 8 threads.

5.3 Analysis of Test Results

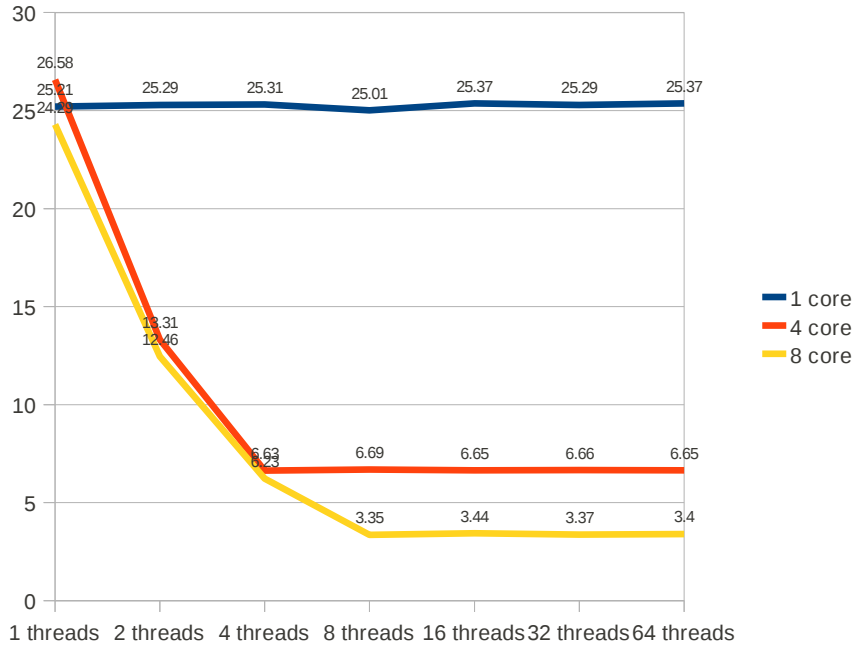


Figure 15: Test results for Amazon Virtual Machines

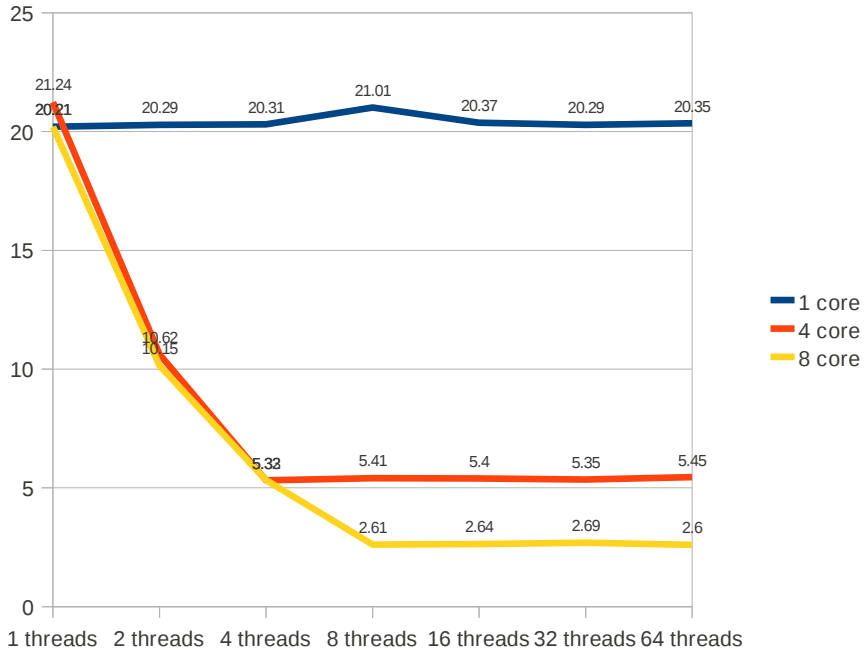


Figure 16: Test results for physical machine

The Figure 15 and Figure 16 shows that output given by Amazon virtual machine are similar to the physical machines of similar configuration. The output generated by the physical machines is slightly faster then Amazon virtual machines but there are other factors such as cache and processor speed that may be responsible for the variation. The results suggest that virtual machines from the cloud can be used as replacement for the physical resources without sacrificing much performance.

6 Uses

This application can be used as a lab environment for students. The students will have user privileges. They will be provided with user-name and password to use the service. They can start, terminate, or connect to the virtual instance generated by this EC2Lab application. The virtual instances will be used to experiment with parallelism and understand cloud infrastructure layer concepts. The administrator will monitor the instances started by the students and terminate unused instances.

This application can also be used by small organization who want to procure the physical resources on-demand from the cloud, and work with high performance machines on pay-per-use basis. The administrator can easily setup and add new machines that can be used by the employees for development and testing. The administrator can monitor the instances and terminate instances, if required. The employees can use the user privileges to control the instances and connect to the instances.

This application will be helpful to many users as they can easily procure the resources on-demand without going through the account creation, authentication, and setting up the machine images. It will help them to:

- understand the cloud concept and its elastic capability,
- Procure and release the cloud infrastructure resources easily

7 Conclusion and Future Enhancements

This project successfully creates a lab environment for users to work with range of multi-core machines. It enables users to instantiate Amazon Machine Images on-demand using this Software-as-a-Service(SaaS) web application. It helps them to easily start, monitor, and terminate the instances via an intuitive web interface, and provides an excellent learning platform to explore physical resource serving capabilities of the cloud.

The tests are conducted on virtual machines procured from Amazon and similar physical machine. The results support that cloud resources provide similar performance, and can be used as a replacement for actual physical machines.

Future Enhancements:

This project can be enhanced further to include two new features:

1. In-built shell
2. Cloud provider independence

1. **In-built Shell:** An in-build interactive PHP shell can be included in this application. Currently an external shell terminal is required to execute the SSH command for connecting to the virtual instances. Providing an in-built shell functionality would eliminate the need of an external shell.

2. **Cloud provider independence:** The physical resources are procured from Amazon, which is a leading infrastructure provider. This project can be enhanced to make the application provider independent. New layer can be added to enable the application to procure resources from any public infrastructure provider.

Appendix A: References

[1] Adaptive Computing(2010). *Moab Adaptive Computing Suite.* Retrieved on Nov 1, 2010 from http://www.adaptivecomputing.com/products/moab-adaptive-computing-suite.php

[2] Amazon Web Services(2010). Retrieved on November 10, 2010 from http://aws.amazon.com/

[3] Amazon Web Services (2010). *Amazon Elastic Cloud Compute: Getting started Guide.* (2010). Retrieved on Nov 09, 2010 from http://docs.amazonwebservices.com/AWSEC2/2007-08-29/GettingStartedGuide/

[4] Amazon Web Service(2011). *Amazon PHP SDK.* Retrieved on April 21, 2011 from website: http://docs.amazonwebservices.com/AWSSDKforPHP/latest/

[5] AKIOKA, S. AND MURAOKA, Y. 2010. HPC Benchmarks on Amazon EC2. In *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on,* Anonymous , 1029-1034.

[6] BAUN, C. AND KUNZE, M. 2009. Building a private cloud with Eucalyptus. In *E-Science Workshops, 2009 5th IEEE International Conference on,* Anonymous , 33-38.

[7] CHUNYE GONG, JIE LIU, QIANG ZHANG, HAITAO CHEN AND ZHENGHU GONG. 2010. The Characteristics of Cloud Computing. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on,* Anonymous , 275-279.

[8] Information week(2011).*Cloud Revenue*.Retrived on July 2011 from http://www.informationweek.com/news/global-cio/interviews/231000596

[9] CONG WANG, KUI REN, WENJING LOU AND JIN LI. 2010. Toward publicly auditable secure cloud data storage services. *Network, IEEE* 24, 19-24.

[10] COOPER, B. 2010. The Prickly Side of Building Clouds. *Internet Computing, IEEE* 14, 64-67.

[11] DIKAIAKOS, M.D., KATSAROS, D., MEHRA, P., PALLIS, G. AND VAKALI, A. 2009. Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *Internet Computing, IEEE* 13, 10-13. .

[12] DONG, H., HAO, Q., ZHANG, T. AND ZHANG, B. 2010. Formal Discussion on Relationship between Virtualization and Cloud Computing. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on,* Anonymous , 448-453.

[13] Eucalyptus, 2010. Retrieved on April 21, 2011 from http://www.eucalyptus.com/

[14] GAIKWAD, M. (2010). Understanding the Cloud Using Amazon EC2 and S3.

[15] Gartner (2011). Retrieved on Nov 2011 from http://www.gartner.com/it/page.jsp?id=1526414

[16] GUOHUI WANG AND NG, T.S.E. 2010. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *INFOCOM, 2010 Proceedings IEEE,* Anonymous , 1-9.

[17] Illinicloud org. Retrieved on Nov 10,2011 from http://www.illinicloud.com/

[18] JACKSON, K.R., RAMAKRISHNAN, L., MURIKI, K., CANON, S., CHOLIA, S., SHALF, J., WASSERMAN, H.J. AND WRIGHT, N.J. 2010. Performance Analysis of High Performance Comput

ing Applications on the Amazon Web Services Cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Anonymous , 159-168.

[19] KHAN, K.M. AND MALLUHI, Q. 2010. Establishing Trust in Cloud Computing. *IT Professional* 12, 20-27.

[20] KAEFER,G (2010). Cloud Computing Architecture. Retrived on Nov 10,2011 from http://www.sei.cmu.edu/library/assets/presentations/Cloud%20Computing%20Architecture%20-%20Gerald%20Kaefer.pdf

[21] LINGLI FU AND TAO CHEN. 2010. Building enterprise application based on cloud computing. In *E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on*, Anonymous , 534-537.

[22] MICHAEL ARMBRUST, ARMANDO FOX, REAN GRIFFITH, ANTHONY D. JOSEPH, RANDY H. KATZ. 2009. Above the Clouds: A Berkeley View of Cloud Computing.

[23] MINQI ZHOU, RONG ZHANG, DADAN ZENG AND WEINING QIAN. 2010. Services in the Cloud Computing era: A survey. In *Universal Communication Symposium (IUCS), 2010 4th International*, Anonymous , 40-46.

[24] MOORE, G.A. 2002. *Crossing the Chasm.* Harper Collins Publishers.

[25] *Mobile Industry & Business Issues. (2010)*. Retrieved Nov 10, 2010 from http://mobilepov.wordpress.com/

[26] OpenMP C and C++ Application Program Interface. Retrieved on Nov 10,2011 from http://www.openMP.org

[27] PALLIS, G. 2010. Cloud Computing: The New Frontier of Internet Computing. *Internet Computing, IEEE* 14, 70-73. .

[28] Saving a Customised Linux Amazon Instance (EC2 and S3) (2009). Retreived on November , 2011 from http://robrohan.com/2009/01/30/saving-a-customised-linux-amazon-instance-ec2-and-s3/

[29] SCHAFFNER, J., JACOBS, D., ECKART, B., BRUNNERT, J. AND ZEIER, A. 2010. Towards enterprise software as a service in the cloud. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, Anonymous , 52-59.

[30] SHUFEN ZHANG, SHUAI ZHANG, XUEBIN CHEN AND SHANGZHUO WU. 2010. Analysis and Research of Cloud Computing System Instance. In *Future Networks, 2010. ICFN '10. Second International Conference on*, Anonymous , 88-92.

[31] STANTCHEV, V. 2009. Performance Evaluation of Cloud Computing Offerings. In *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on*, Anonymous , 187-192.

Appendix B: Test Program Sample Code

```
#include<stdio.h>
#include<omp.h>
#include<time.h>
long long num_steps = 1000000000;
double step;
int main(int argc, char * argv[])
{
        double x,pi,sum=0.0;
        int i;
        step = 1.0/(double)num_steps;
        omp_set_num_threads(4); //This number is changed to vary the threads
        #pragma omp parallel for private(x) reduction(+:sum)
                for(i=0;i<num_steps:i++) {
                        x = (i+0.5) * step;
                        sum = sum + 4.0/(1.0 + x*x);
                }
        pi = sum*step;
        printf("The value of PI is %15.12f \n",pi);
}
```

Appendix C: Project Source Code

Folder: Model
File name: AccountOperations.php –

```
<?php
require_once("DatabaseAccess.php");
/**
 * Acctoperations
 *
 * @package  project_EC2Lab/models
 * @author Manisha Gaikwad
 * @version 2011
 *  * Model class to perform all database related operations using
 * DatabaseAccess.php functions
 */
class AccountOperations {
   /**
    * DataOperations::checkAccount()
    *
    * @param mixed $username
    * @param mixed $password
    * @return
    */
   public function checkAccount($username,$password,$type) {
      //Check Username & Password for signin
      $database = new DatabaseAccess();
      $database->open();
          $query="select count(*) from User_info where user_name='{$username}' and
password = '{$password}' and type = '{$type}'";
         $results = $database->ExecuteQuery($query)  or die("Query failed with error:
".mysql_error());
      $row = mysql_fetch_array($results);
      $database->close();
      // if count = 1 i.e. entry found return appropriate status
      if ($row['count(*)'] == 1)
      {
         return true;
      } else {
         return false;
      }

   }
}
```

```
?>
```

File: AjaxStart.php

```
<?php
/*
%*************************************************************************
**********************%*/

// SETUP
        // Enable full-blown error reporting. http://twitter.com/rasmus/status/7448448829
        error_reporting(-1);
        // Set HTML headers
        header("Content-type: text/html; charset=utf-8");
        // Include the SDK
        require_once ('../sdk/sdk/sdk.class.php');
        require_once('UserAdminDataOperations.php');
/*
%*************************************************************************
**********************%*/

// Instantiate the class
$ec2 = new AmazonEC2();
if (isset($_GET['id']))
        {       $id = $_GET['id'];
                  while(1)
                  {
                          $describe = $ec2->describe_instances(array('InstanceId' => $id ));
                          $state = $describe->body->reservationSet->item->instancesSet-
>item->instanceState->name[0];
                          if( $state == 'running')
                          {
                                  $instance_data['dnsname']          =          $describe->body-
>reservationSet->item->instancesSet->item->dnsName[0];
                                  $instance_data['keyname']          =          $describe->body-
>reservationSet->item->instancesSet->item->keyName[0];
                                  $instance_data['instancetype']       =       $describe->body-
>reservationSet->item->instancesSet->item->instanceType[0];
                                  $instance_data['launchtime']       =       $describe->body-
>reservationSet->item->instancesSet->item->launchTime[0];
                                  $instance_data['virtualizationType']   =   $describe->body-
>reservationSet->item->instancesSet->item->virtualizationType[0];
```

```
                                $instance_data['ami_id'] = $describe->body->reservationSet->item->instancesSet->item->imageId[0];
                                $instance_data['instance_id'] = $id;
                                $instance_data['status'] = 2;//1-pending, 2-running ,3-terminated

                                $instance_data['username'] = $_REQUEST['username'];
                                $model = new UserAdminDataOperations();
                                $model->saveUserInstance($instance_data);
                                $dnsname = $instance_data['dnsname'];
                            echo "Instance with ID: $id successfully started<br>\nconnect to dns:  $dnsname using putty or SSH<br>";
                                 echo "\n<br>For SSH connection: <b>ssh -i keypair1.pem ec2-user@$dnsname</b>";
                                break;
                        }
                        // sleep for 2 seconds
                        sleep(2);
                }
        }
 ?>
```

File: AjaxTerminate.php

```
<?php
/*
%************************************************************************
**********************%*/
// SETUP
        // Enable full-blown error reporting. http://twitter.com/rasmus/status/7448448829
        error_reporting(-1);
        // Set HTML headers
        header("Content-type: text/html; charset=utf-8");
        // Include the SDK
        require_once ('../sdk/sdk/sdk.class.php');
        require_once('UserAdminDataOperations.php');
/*
%************************************************************************
**********************%*/
// Instantiate the class
        $ec2 = new AmazonEC2();
//terminate instances
        $instances = explode(",",$_REQUEST['instances']);
```

```
        while(1)
                {
                        $describe       =       $ec2->describe_instances(array('InstanceId'
=>$instances ));
                        //var_dump($describe);
                        if(count($instances) >= 1) {
                                $stateAllTerminate = true;

                        }
                        for($i=0;$i<count($instances);$i++) {
                                $state      =      $describe->body->reservationSet->item[$i]-
>instancesSet->item->instanceState->name[0];
                                if($state != 'terminated') {
                                   $stateAllTerminate = false;
                                   break;
                                }
                        }
                if( $stateAllTerminate == true)
                        {
                                $model = new UserAdminDataOperations();
                                $model->removeUserInstance($instances);
                                                        echo      "Instance      with      ID:".
$_REQUEST['instances'] ."successfully terminated!!<br>";
                                break;
                        }
                        // sleep for 2 seconds
                        sleep(2);
                }
?>
```

File: Config.php

```
<?php
class db_username_pass{
public static  $username="root";
public static  $password="manisha";
public static  $keypair="keypair1";
}
?>
```

File: DataAccess.php

```
<?php
```

```
/**
 * DatabaseAccess
 *
 * @package  project_EC2Lab/model
 * @author Manisha Gaikwad
 *
 * Model class to access Mysql Database tables
 *
 * Contains:
 * DbConnect, DataAccess, Insert, Select etc.. utilities
 */

require_once("config.php");
class DatabaseAccess{
   private $con="";
   /**
    * DatabaseAccess::open()
    *
    * @return
    */
   //require_once ('config/config.php');
   public function open(){
   // connects to database "urlService"
   $this->con =
            mysql_connect("localhost",db_username_pass::$username,db_username_pass::
$password);
   if (!$this->con)
      die('Could not connect: ' . mysql_error());
   mysql_select_db("EC2Lab_DB", $this->con);
   }

   /**
    * DatabaseAccess::close()
    *
    * @return
    */
   public function close() {                                                    //          closes
connection when object dies
      mysql_close($this->con);
   }

   /**
    * DatabaseAccess::ExecuteQuery()
    *
```

```
    * @param mixed $queryString
    * @return
    */
   public function ExecuteQuery($queryString)        {                                    //
execute queries and return the result array
       $results = mysql_query($queryString);
       return $results;
   }
}
?>
```

File: InstanceOperations.php

```
<?php

class InstanceOperations {
   public function InstanceOperations() {
                    // Set HTML headers
                    header("Content-type: text/html; charset=utf-8");
   }
  /**
    * InstanceOperations::startInstance()
    *
    * @param mixed $username
    * @return
    */
   public function startInstance($username) {
         // Include the SDK
         require_once ('sdk/sdk/sdk.class.php');
         $ami = null; $size= null;
         if(isset($_REQUEST['start1']) &&  $_REQUEST['start1'] == 'start') {
                    $ami  = "ami-6f5c9406";
                    $size = "m1.small";
         } elseif(isset($_REQUEST['start2']) && $_REQUEST['start2'] == 'start') {
                    $ami  = "ami-221fec4b";
                    $size = "m1.xlarge";
         } elseif( isset($_REQUEST['start3']) && $_REQUEST['start3'] == 'start') {
                    $ami  = "ami-221fec4b";
                    $size = "c1.xlarge";
         }
         if($ami != null || $size != null)
         {
                    // Instantiate the class
                    $ec2 = new AmazonEC2();
```

```
                $response = $ec2->run_instances($ami, 1, 1, array( 'KeyName' => 'keypair1', 'SecurityGroup' => 'default', 'InstanceType' => $size));
                if($response->isOK() == true) {
                        $result['success'] = $response->isOK();
                        $result['id'] = $response->body->instancesSet->item->instanceId[0];
                } else {
                        $result['success'] = false;
                        $result['info'] = "Instance cannot be created";
                }
        }
        else
        {
                $result['success'] = false;
                $result['info'] = "Instance cannot be created";
        }
        return $result;
   }
  public function describe_instances() {
        require_once ('sdk/sdk/sdk.class.php');
        $ec2 = new AmazonEC2();
        $response = $ec2->describe_instances();
      return $response;
  }
  public function terminateInstances($instances) {
        require_once ('sdk/sdk/sdk.class.php');
        $ec2 = new AmazonEC2();
        $response = $ec2->terminate_instances($instances);
        if($response->isOK() == true) {
                        $result['success'] = true;
                        $result['instances'] = implode(',',$instances);
        } else {
                        $result['success'] = false;
        }
        return $result;
  }
}
?>
```

File: UserAdminDataOperations.php

```
<?php
require_once("DatabaseAccess.php");
require_once ('config.php');
```

```
require_once("InstanceOperations.php");

class UserAdminDataOperations {
   /**
    * DataOperations::getUserData()
    *
    * @param mixed $username
    * @param mixed $password
    * @return
    */
   public function getUserData($username) {
      //Check Username & Password for signin
      $database = new DatabaseAccess();
      $database->open();
        $query="select instance_id,status,type,virtualization,connect_info,launchtime from
instance_table where user_name = '{$_SESSION['username']}'";
      $results = $database->ExecuteQuery($query);
      $i=0;
         $userData = null;
      $no_rows = mysql_num_rows($results);
      //get data from instance_table statistics
      if ($no_rows != 0) {
            while ($row = mysql_fetch_array($results)) {
               $userdata_row['instance_id'] = $row['instance_id'];
               $userdata_row['status'] = $row['status'];
            $userdata_row['type'] = $row['type'];
                  $userdata_row['virtualization'] = $row['virtualization'];
                  $userdata_row['connect_info'] = $row['connect_info'];
                  $userdata_row['launchtime'] = $row['launchtime'];
                  $userData[$i]=$userdata_row;
            $i++;
            }
      }
      $database->close();
      return $userData;
   }
   /**
    * DataOperations::getAdminData()
    *
    * @param mixed $username
    * @param mixed $password
    * @return
    */
   public function getAdminData($username) {
```

```
	//Check Username & Password for signin
	$instanceOp = new InstanceOperations();
	   $adminData = null;
	$response = $instanceOp->describe_instances();
	//var_dump($response);
	   // Extract items from the response
	   if(isset($response->body->reservationSet->item)) {
		$instances = $response->body->reservationSet->item;
	// Get user data from instance_id's
		$i=0;
		$userData = $this->getUsersforInstances($instances);
		//var_dump($userData);
		$i=0;
		foreach($instances as $row) {
			$adminData_row['instance_id'] = $row->instancesSet->item->instanceId[0];
			$adminData_row['ami_id'] = $row->instancesSet->item->imageId[0];
			//get username for the instance
			if(isset($userData)) {
				foreach($userData as $user_row) {
					if($user_row['instance_id'] ==$adminData_row['instance_id']) {
						$adminData_row['user_name'] = $user_row['user_name'];
						break;
					}
				}
			}
			  $adminData_row['status'] = $row->instancesSet->item->instanceState->name[0];
		$adminData_row['type'] = $row->instancesSet->item->instanceType[0];
			$adminData_row['virtualization'] = $row->instancesSet->item->virtualizationType[0];
			$adminData_row['dns_name'] = $row->instancesSet->item->dnsName[0];
			$adminData_row['launch_time'] = $row->instancesSet->item->launchTime[0];
			$adminData_row['keypair'] = $row->instancesSet->item->keyName[0];
			$adminData[$i]=$adminData_row;
		$i++;
		   }
	}
```

```
        return $adminData;
    }

    public function saveUserInstance($data) {
        $database = new DatabaseAccess();
        $database->open();
        $query="INSERT INTO instance_table
                VALUES ('{$data['instance_id']}', '{$data['username']}',
                    '{$data['status']}','{$data['instancetype']}','{$data['keyname']}',
                    '{$data['virtualizationType']}','{$data['dnsname']}',
                    '{$data['launchtime']}','{$data['ami_id']}')";
        $results = $database->ExecuteQuery($query);
        $database->close();
    }

    public function removeUserInstance($instances) {
        $database = new DatabaseAccess();
        $database->open();
        $inClause = null;
        foreach($instances as $key => $value) {
                    if($inClause == null) {
                                $inClause = "'{$value}'";
                        } else {
                                $inClause = $inClause . ",'{$value}'";
                        }

            }
        $query="DELETE FROM instance_table where instance_id in (".$inClause.")";
        $results = $database->ExecuteQuery($query);
        $database->close();
    }
    public function getUsersforInstances($instances) {
        $database = new DatabaseAccess();
        $database->open();
        $inClause = null;
            foreach($instances as $row) {
                    if($inClause == null) {
                                $inClause = "'{$row->instancesSet->item-
>instanceId[0]}'";
                        } else {
                                $inClause = $inClause . ",'{$row->instancesSet->item-
>instanceId[0]}'";
                        }
            }
```

```
        $query="select instance_id,user_name from instance_table where instance_id in ($inClause)";
      $results = $database->ExecuteQuery($query);
      $i=0;
         $userData = null;
      $no_rows = mysql_num_rows($results);
      //var_dump($results);
      //get data from instance_table statistics
      if ($no_rows != 0) {
            while ($row = mysql_fetch_array($results)) {
               $userdata_row['instance_id'] = $row['instance_id'];
               $userdata_row['user_name'] = $row['user_name'];
                  $userData[$i]=$userdata_row;
            $i++;
            }
      }
      $database->close();
      return $userData;
   }


   /*public function terminateInstances($user_name,$user_type,$instances) {
         $instanceOp = new InstanceOperations();
      $response = $instanceOp->terminate_instances($instances);
   }*/
 }
?>
```

Folder: View

File: AdminVew.php

```
<?php
/**
 * Display
 *
 * @package Project_Ec2Lab/Views
 * @author Manisha Gaikwad
 * @version 2011
 *
 * View to display pages
 */

class AdminView {
```

```
/**
 * Display::render()
 *
 * @param mixed $data
 * @return
 */

public function render($userData,$navLinks){
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <title>Ec2Lab</title>
        <meta name="Description" content="Ec2Lab - SaaS for Amazon Ec2
Instanciation" />
        <meta name="robots" content="all, index, follow" />
        <meta name="distribution" content="global" />
        <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />
<script language="javascript" type="text/javascript">
<!--
//Browser Support Code
        function getCheckedRows(boxes){
                var checkBoxStr=null;
                for(i=0;i<boxes.length;i++) {
                        if(boxes[i].checked == true) {
                                if(checkBoxStr == null) {
                                        checkBoxStr = boxes[i].value;
                                } else {
                                        checkBoxStr = checkBoxStr + ',' + boxes[i].value;
                                }
                        }
                }
                //if single entry then
                if(checkBoxStr==null && boxes.checked == true) {
                        checkBoxStr = boxes.value;
                }
                document.getElementById('boxesChecked').value = checkBoxStr;
        }
-->
</script>
</head>
```

```
<body class="home">
<div id="container">
        <div id="header">

                        <img        id="logo_width"        title="EC2Lab"        alt="EC2Lab"
src="./images/logo.jpg" />
        </div>
        <div id="navigation">
                <ul>
                        <li><a href="<?php echo $navLinks['links'][0];?>"><?php echo
$navLinks['links'][1];?></a></li>
                        <li><a href="<?php echo $navLinks['links'][2];?>"><?php echo
$navLinks['links'][3];?></a></li>
                </ul>
        </div>
        <form action="./index?c=ATerminateOp" method="post" name="adminview">
        <div id="content-container">
<!-- Second part instance information-->
        <div> <h2> Summary of Running Instances<h2></div>
        <?php
        if (isset($userData)) {

                echo "<input type='submit' id='delete' name='terminate' value='Terminate
Instances' onClick='javascript:getCheckedRows(document.adminview.boxes)'/>";
                echo "<table border='1'>
                <tr>
                <th></th>
                <th>Status</th>
                <th>Instance ID</th>
                <th>Ami        ID</th>
                <th>User Name</th>
                <th>Type</th>
                <th>Launch Time</th>
                <th>Dns name</th>
                <th>Key Name</th>

                <th>Virtualization</th>                </tr>";
                $i=0;
        foreach($userData as $row) {
                echo "<tr>";
                        echo    "<td>"    .    "<input    type='checkbox'    name='boxes'
value='{$row['instance_id']}'/>"."</td>";
                        echo "<td>" . $row['status'] . "</td>";
                        echo "<td>" . $row['instance_id'] . "</td>";
```

```
                        echo "<td>" . $row['ami_id'] . "</td>";
                        echo "<td>" . $row['user_name'] . "</td>";
                        echo "<td>" . $row['type'] . "</td>";
                        echo "<td>" . $row['launch_time'] . "</td>";
                        echo "<td>" . $row['dns_name'] . "</td>";
                        echo "<td>" . $row['keypair'] . "</td>";
                        echo "<td>" . $row['virtualization'] . "</td>";
                        echo "</tr>"; $i++;
                }
                echo "</table>";
                echo "<br><br>";
        } else {
                echo "<br>";
                echo "<h4> No Instances Found!</h4>";
                echo "<br><br>";
        }
?>
<!-- Second part instance information-->
<input type="hidden" id="boxesChecked" name="boxesChecked"/>
        <form>
        </div>          <div id="footer">
                Copyright © EC2Lab, 2011
        </div>
</div>
</body>
</html>
<?php } }?>
```

File: Display.php

```
<?php
/**
 * Display
 *
 * @package Project_Ec2Lab/Views
 * @author Manisha Gaikwad
 * @version 2011 *
 * View to display pages
 */
class Display {

/**
```

```
 * Display::render()
 *
 * @param mixed $data
 * @return
 */
public function render($navLinks,$status){
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <title>Ec2Lab</title>
        <meta name="Description" content="Ec2Lab - SaaS for Amazon
Ec2Instanciation" />
        <meta name="robots" content="all, index, follow" />
        <meta name="distribution" content="global" />
        <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />
</head>
<body class="home">
<div id="container">
        <div id="header">

                        <img id="logo_width" title="EC2Lab" alt="EC2Lab"
src="./images/logo.jpg" />

        </div>
        <div id="navigation">
                <ul>
                        <li><a href="<?php echo $navLinks['links'][0];?>"><?php echo
$navLinks['links'][1];?></a></li>
                        <li><a href="<?php echo $navLinks['links'][2];?>"><?php echo
$navLinks['links'][3];?></a></li>
                </ul>
        </div>
        <div id="content-container">
           <form method="post" action="index.php?c=signin&v=signin">
              <div id="content">
                        <div id="login_div_space1"><h2> Login </h2></div>
                     <div id="login_div_space2"> USERNAME </div>
                        <div id="login_div_space2"> <input type="text"
name="username" /> </div>
                        <div id="login_div_space2"> PASSWORD </div>
```

```
                    <div id="login_div_space2"> <input type="password" name="password" /> </div>
                    <div id="login_div_space2"> TYPE </div>
                    <div id="login_div_space2">
                        <select name="type">
                          <option value="0">User</option>
                          <option value="1">Admin</option>
                        </select>
                    </div>
                    <div id="login_div_space1" class="login_div_space3"><input type="submit" name="Submit" value="Submit"/></div>
                    <div id="error"> <?php echo $status['error'];?>  </div>

            </div>
        </form>
        <div id="aside">
            <img id="logo_fleft" title="EC2Lab" alt="EC2Lab" src="./images/image1.jpg" />
            <h3>
                About EC2Lab
            </h3>
            <p>
                "EC2Lab" is a Software-as-a-Service (SaaS) web application which is created on top of Infrastructure-as-a-Service (IaaS) layer. It helps
                users to quickly create, connect, and terminate virtual machine instances. Thus, providing a quick facility to create and handle amazon
                instances.
            </p>
            <p>
                This web application harnesses the elastic capability of the Amazon's IaaS layer. It is created using Amazon Web Service and PHP
                SDK, and will serve as a learning platform to explore the capabilities of the cloud. It will help to understand cloud's infrastructure layer and elastic capability. The users just need the login credentials to experiment with Amazon Infrastructure layer.
            </p>
        </div>
        <div id="footer_section">
            <div id="col1">
            <a href="http://aws.amazon.com/"><h2>Amazon Web services</h2></a>
            <p></p>
            </div>
            <div id="col2">
            <a href="http://aws.amazon.com/sdkforphp/"><h2>PHP SDK</h2></a>
```

```
                <p></p>
                </div>
                <div id="col1">
                <a href="http://aws.amazon.com/ec2/"><h2>Amazon EC2</h2></a>
                <p></p>
                </div>
        </div>
        <div id="footer">
                Copyright © EC2Lab, 2011
        </div>
        </div>
</div>
</body>
</html>
<?php } }?>
```

File:InprogressView.php

```
<?php
class InprogressView {

public function render($result,$navLinks,$usertype,$terminate){
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <title>Ec2Lab</title>
        <meta name="Description" content="Ec2Lab - SaaS for Amazon Ec2
Instanciation" />
        <meta name="robots" content="all, index, follow" />
        <meta name="distribution" content="global" />
        <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />

<script language="javascript" type="text/javascript">
<!--
//Browser Support Code
function ajaxFunction(){
        var ajaxRequest;  // The variable that makes Ajax possible!

        try{
```

```
		// Opera 8.0+, Firefox, Safari
		ajaxRequest = new XMLHttpRequest();
	} catch (e){
		// Internet Explorer Browsers
		try{
			ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
		} catch (e) {
			try{
				ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
			} catch (e){
				// Something went wrong
				alert("Your browser broke!");
				return false;
			}
		}
	}
	// Create a function that will receive data sent from the server
	ajaxRequest.onreadystatechange = function(){
		if(ajaxRequest.readyState == 4){
		id = document.getElementById("text_main");
			//id.style.text-align = "left";
			id.innerHTML = ajaxRequest.responseText;
		button = document.getElementById("dashboard");
			button.disabled = false;
			document.getElementById("loadinggif").style.visibility = 'hidden';
		}
	}
	terminate = document.getElementById('terminate').value;
	//alert(terminate);
	//use with terminate instance
	if(terminate == true) {
		//send the instance id array
		params = "username=" + document.getElementById('username').value + "&terminate=true" +
"&instances=" + document.getElementById('instances').value;
		ajaxRequest.open("POST", "model/AjaxTerminate.php", true);
		ajaxRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded")
		ajaxRequest.send(params);

	} else { //used with start instances
		params = "?id=" + document.getElementById('instanceId').value + "&username=" + document.getElementById('username').value + "&terminate=false";
```

```
                //alert(params);
                ajaxRequest.open("GET", "model/AjaxStart.php"+params, true);
                //ajaxRequest.setRequestHeader("Content-type","application/x-www-
form-urlencoded")
                ajaxRequest.send(null);
        }
}

//-->
</script>
</head>

<?php
//check of onload should be called
if($result['success'] == true) {
        $onload = "javascript:ajaxFunction()";
} else {
        $onload = "";
}

if ($usertype == "admin") {
        $action = "./index.php?c=admin";
} else {
        $action = "./index.php?c=user";
}

?>
<body class="home" onLoad="<?php echo $onload;?>" >
<div id="container">
        <div id="header">

                        <img id="logo_width" title="EC2Lab" alt="EC2Lab"
src="./images/logo.jpg" />

        </div>

        <div id="navigation">
                <ul>
                        <li><a href="<?php echo $navLinks['links'][0];?>"><?php echo
$navLinks['links'][1];?></a></li>
                        <li><a href="<?php echo $navLinks['links'][2];?>"><?php echo
$navLinks['links'][3];?></a></li>
                </ul>
        </div>
```

```
<form method="post" action="<?php echo $action;?>">
<div id="content-container">
<div id = "header">
<div id="header_right1" style="text-align: center;"><br></br>
<?php if($terminate == true) {
echo "<h2 id='head_sub'>Terminating Amazon AMI....</h2>";
} else {
echo "<h2 id='head_sub'>Instantiating Amazon AMI....</h2>";
}
?>
</div>
</div>
<div id ="main_start1" style="text-align: center;">
<?php if($result['success'] == true) {
echo "<img src='images/loading_bar.gif' id='loadinggif'/>";
echo "<div id = 'text_main'>";
if($terminate == true) {
echo "<br><br>
Please wait while the instances are being terminated";
}else {
echo "<br><br>
Please wait while the machine is being instanciated";
}
?>

</div>
<br> <br>
<input type="submit" id="dashboard" name="DashBoard" value="Go Back to Dashboard" disabled="true"/>
<? } else {?>
<img src="images/error.png" id="loadinggif"/>
<div id = "error">
<br><br>
Error: Operation cannot be completed
</div>
<br> <br>
<input type="submit" id="dashboard" name="DashBoard" value="Go Back to Dashboard" disabled="true"/>
<?php }?>
</div>
<div id="footer">
Copyright © EC2Lab, 2011
</div>
</div>
```

```
<input type="hidden" name="instanceId" id="instanceId" value="<?php echo $result['id'];?>"/>
<input type="hidden" name="username" id="username" value="<?php echo $_SESSION['username'];?>"/>
<input type="hidden" name="terminate" id="terminate" value="<?php echo $terminate;?>"/>
<input type="hidden" name="instances" id="instances" value="<?php echo $result['instances'];?>">
</form>

  </body>
</html>

<?php
} }
?>
```

File:InstanceDetails.php

```
<?php
/**
 * Display
 *
 * @package Project_Ec2Lab/Views
 * @author Manisha Gaikwad
 * @version 2011
 *
 * View to display pages
 */
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <title>Ec2Lab</title>
        <meta name="Description" content="Ec2Lab - SaaS for Amazon Ec2 instanciation" />
        <meta name="robots" content="all, index, follow" />
        <meta name="distribution" content="global" />
        <link rel="stylesheet" href="../css/style.css" type="text/css" media="screen" />
</head>
```

```
<body class="home">
<div id="container">
        To connect to the instance type the bellow command at the shell -<br><br>
        <?php echo "<b>"."ssh -i keypair1.pem ec2-user@".$_REQUEST['connect']."</b>";?>
</div>
</body>
</html>
<?php ?>
```

File:UserView.php

```
<?php
/**
 * Display
 *
 * @package Project_Ec2Lab/Views
 * @author Manisha Gaikwad
 * @version 2011
 *
 * View to display pages
 */
class UserView {


/**
 * Display::render()
 *
 * @param mixed $data
 * @return
 */
public function render($userData,$navLinks){
?>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <title>Ec2Lab</title>
        <meta name="Description" content="Ec2Lab - SaaS for Amazon Ec2 Instanciation" />
        <meta name="robots" content="all, index, follow" />
        <meta name="distribution" content="global" />
```

```
        <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />
<script language="javascript" type="text/javascript">
<!--
//Browser Support Code
        function getCheckedRows(boxes){
                var checkBoxStr=null;
                for(i=0;i<boxes.length;i++) {
                        if(boxes[i].checked == true) {
                                if(checkBoxStr == null) {
                                        checkBoxStr = boxes[i].value;
                                } else {
                                        checkBoxStr = checkBoxStr + ',' + boxes[i].value;
                                }
                        }
                }
                //if single entry then
                if(checkBoxStr==null && boxes.checked == true) {
                        checkBoxStr = boxes.value;
                }
                document.getElementById('boxesChecked').value = checkBoxStr;
                document.getElementById('terminateOP').value = "true";
        }
-->
</script>

</head>
<body class="home">
<div id="container">
        <div id="header">

                        <img id="logo_width" title="EC2Lab" alt="EC2Lab"
src="./images/logo.jpg" />
        </div>
        <div id="navigation">
                <ul>
                        <li><a href="<?php echo $navLinks['links'][0];?>"><?php echo
$navLinks['links'][1];?></a></li>
                        <li><a href="<?php echo $navLinks['links'][2];?>"><?php echo
$navLinks['links'][3];?></a></li>
                </ul>
        </div>
        <form action="./index?c=UInstanceOp" method="post" name="userview">
        <div id="content-container">
<!-- First Part with instance list start-->
```

```
        <div id = "header">
                                <div id="header_right"><br><h4>Welcome <?php echo
$_SESSION['username'];?> !<h4>
              <h2>List of AMI's</h2>Instanciate AMI by pressing start button
            </div>
        <div id ="main">

                 <table cellspacing="4" cellpadding="4" border="1">
                   <tr>
                      <th></th>
                         <th>Instance type</th>
                         <th>Description</th>
                   </tr>

                 <tr>
                  <td><input align="center" type="submit" id="start1" name="start1" value
="start"> </td>
                   <td>Small</td>
                   <td>1 Core 1 ECU</td>
                 </tr>
                 <tr>

                         <td><input       align="center"       type="submit"       id="start2"
name="start2" value ="start"> </td></td>
                         <td>Medium</td>
                         <td>4 Core 8 ECU</td>

             </tr>
                <tr>
                         <td>  <input  align="center"  type="submit"  id="start3"
name="start3" value ="start"> </td></td>

                         <td>Large</td>
                         <td>8 Core 20 ECUs</td>

                    </tr>
                 </table>
        </div>
<!-- First Part end-->
<!-- Second part instance information-->
<div> <h2> Summary of Instances<h2></div>
<?php
        if (isset($userData)) {
```

```
        echo "<input type='submit' id='delete' name='terminate' value='Terminate Instances'
onClick='javascript:getCheckedRows(document.userview.boxes)'/>";
                    echo "<table border='1'>
                    <tr>
                    <th></th>
                    <th>Status</th>
                    <th>Instance ID</th>
                    <th>Type</th>
                    <th>Virtualization</th>
                    <th>Launch Time</th>
                    <th>Connection</th>
                    </tr>";
                    foreach($userData as $row) {
                      echo "<tr>";
                       echo "<td>" . "<input type='checkbox' name='boxes'
value='{$row['instance_id']}'/>"."</td>";
                      if($row['status'] == 2) {
                              echo "<td> running </td>";
                      } else {
                              echo "<td>" . "</td>";
                      }
                      echo "<td>" . $row['instance_id'] . "</td>";
                      echo "<td>" . $row['type'] . "</td>";
                      echo "<td>" . $row['virtualization'] . "</td>";
                      echo "<td>" . $row['launchtime'] . "</td>"; ?>

                          <td> <input type='button' name='connect' value=' Connect '
onclick="javascript: window.open('view/InstanceDetails.php?<?php echo
'c=user&connect='. $row['connect_info'];?
>','_blank','toolbar=no,titlebar=no,location=no,status=no,menubar=no,scrollbars=no,resiz
able=no,width=600,height=100'); return false; "/> </td>
                      <?php echo "</tr>";
                    }
                    echo "</table>";
                    echo "<br><br>";
          } else {
                    echo "<h4>No records found!!</h4>";
          }
?>
<!-- Second part instance information-->
<input type="hidden" id="boxesChecked" name="boxesChecked"/>
<input type="hidden" id="terminateOP" name="terminateOP" value="false"/>
          <form>
          </div>
```

```
        <div id="footer">
                Copyright © EC2Lab, 2011
        </div>


</div>
</body>
</html>
<?php } }?>
```

File: AdminController.php

```
<?php
require_once("view/AdminView.php");
require_once("model/InstanceOperations.php");
require_once("model/UserAdminDataOperations.php");
require_once("view/InprogressView.php");

class AdminController {
      private function get_nav_links()   {
              $navlinks['links'][0] = "./index.php?c=admin&DashBoard=dashboard";
              $navlinks['links'][1] = "DASHBOARD";
              $navlinks['links'][2] = "./index.php?c=default&v=info";
              $navlinks['links'][3] = "INFORMATION";
              return $navlinks;
      }
         public function getdata() {
                $modelOb = new UserAdminDataOperations();
                $adminData = $modelOb->getAdminData($_SESSION['username']);
                return $adminData;
         }
         public function display($adminData) {
                $navlinks = $this->get_nav_links();
                $viewOb = new AdminView();
                $viewOb->render($adminData,$navlinks);
         }
         public function terminateInstances() {
                $modelOb = new InstanceOperations();
                //get the names of selected checkboxes
                $instances = explode(",",$_REQUEST['boxesChecked']);
                $result = $modelOb->terminateInstances($instances);
                $viewOb = new InprogressView();
                $navlinks = $this->get_nav_links();
                $viewOb->render($result,$navlinks,"admin",true);
```

```
        }

}
?>
```

File: DefaultController.php

```
<?php
require_once("view/Display.php");
/**
 * default_controller
 *
 * @package   Project_EC2Lab/Controller
 * @author Manisha Gaikwad
 * @version 2011
 * @access public
 *
 * Home Controller for the index page
 */
class DefaultController {
   private $key = NULL;
   private $url = NULL;
   /**
    * DefaultController::display()
    *
    * @return
    */
   public function display(){

      $navLinks = $this->get_nav_links();
      $viewOb = new Display();
         $status['error'] = "";
      $viewOb->render($navLinks,$status);

   }
   private function get_nav_links()   {
      $navLinks['links'][0] = "./index.php";
      $navLinks['links'][1] = "HOME";
      $navLinks['links'][2] = "./index.php?c=default&v=info";
     $navLinks['links'][3] = "INFORMATION";
         return $navLinks;
   }
}
 ?>
```

File:Signin.php

```
<?php
require_once("model/AccountOperations.php");
require_once("view/Display.php");
require_once("controller/UserController.php");
require_once("controller/AdminController.php");

/**
 * Signin
 *
 * @package   project_EC2lab/Controller
 * @author Manisha Gaikwad
 * @version 2011
 * @access public
 *
 * Signin Controller to perform signin, signout
 * check operations
 */

class Signin {

    /**
     * Signin::signin_call()
     *
     * @return
     */
    public function signin_call()
    {
        $viewOb = new Sign_in();
        $viewOb->render();
    }
    /**
     * Signin::signin_check()
     *
     * @param mixed $username
     * @param mixed $password
     * @return
     */
    public function signin_check($username,$password,$type)
    {
      $model_operations = new AccountOperations();
      $status = $model_operations->checkAccount($username,$password,$type);
      if($status == true ){
          $_SESSION['login'] = true;
```

```
            //$data['nav_link'] = "./index.php?c=signin&v=new_account";
            $_SESSION['username'] = $username;
               $_SESSION['user_type'] = $type;
            if($type == 1) { //admin
                    $controllerOb = new AdminController();
            } else { //user
                    $controllerOb = new UserController();
               }
      $displayData = $controllerOb->getData();
               $controllerOb->Display($displayData);


         } else {
            $_SESSION['login'] = false;
               unset($_SESSION['username']);
            $data['links'][0] = "./index.php";
            $data['links'][1] = "HOME";
            $data['links'][2] = "./index.php?c=default&v=info";
            $data['links'][3] = "INFORMATION";
            $status['error'] = "Login failed try again!!";
               $viewOb = new Display();
            $viewOb->render($data,$status);
         }

      }
      /**
       * Signin::signout()
       *
       * @return
       */
      public function signout()
      {
         $_SESSION['login'] = false;
         session_destroy();
         $viewOb = new SigninSuccess();
         $viewOb->render($data);
      }
}
?>
```

File: UserController.php

```
<?php
require_once("model/UserAdminDataOperations.php");
require_once("view/UserView.php");
require_once("model/InstanceOperations.php");
```

```
require_once("view/InprogressView.php");

class UserController {
	private function get_nav_links()   {
		$navlinks['links'][0] = "./index.php?c=user&DashBoard=dashboard";
		$navlinks['links'][1] = "DASHBOARD";
		$navlinks['links'][2] = "./index.php?c=default&v=info";
		$navlinks['links'][3] = "INFORMATION";
		return $navlinks;
	}
	  public function getdata() {
			$modelOb = new UserAdminDataOperations();
			$userData = $modelOb->getUserData($_SESSION['username']);
			return $userData;
	  }
	  public function display($userData) {
			$navlinks = $this->get_nav_links();
			$viewOb = new UserView();
			$viewOb->render($userData,$navlinks);
	  }
	  public function startInstance() {
			$modelOb = new InstanceOperations();
			$result = $modelOb->startInstance($_SESSION['username']);
			$viewOb = new InprogressView();
			$navlinks = $this->get_nav_links();
			$viewOb->render($result,$navlinks,"user",false);
	  }
	  public function terminateInstances() {
			$modelOb = new InstanceOperations();
			//get the names of selected checkboxes
			$instances = explode(",",$_REQUEST['boxesChecked']);
			$result = $modelOb->terminateInstances($instances);
			$viewOb = new InprogressView();
			$navlinks = $this->get_nav_links();
			$viewOb->render($result,$navlinks,"user",true);
	  }

}

?>
```

File: index.php

```
<?php
/**
```

```
 * @package Project_EC2Lab
 * @author Manisha Gaikwad 00749515
 *
 * index.php
 *  This file acts as first entry point for EC2Lab SaaS and is used to
 *  call appropriate controllers
 */

session_start();
//Choosing the controller
if(!isset($_GET['c'])) {
   $_GET['c'] = "default";
}
if(!isset($_GET['v'])) {
 $_GET['v'] = null;
}
 switch ($_GET['c']) {
    case "signin":
         // lookup for the url
           require_once("controller/Signin.php");
        $signinController = new Signin();
           if ($_GET['v']== "signout" ) {
          //signout
         $signinController->signout_call();
        } else if($_GET['v']== "signin" ){
             //signin
                                                                    $signinController-
>signin_check($_REQUEST['username'],_REQUEST['password'],$_REQUEST['type']);
        }
           break;
    case "admin":
         // lookup for the url
           if( isset($_REQUEST['DashBoard']) && $_SESSION['login'] == true) {
           require_once("controller/AdminController.php");
                   $adminController=new AdminController();
                $displayData = $adminController->getData();
                   $adminController->Display($displayData);
           }
           break;
   case "user":
           if( isset($_REQUEST['DashBoard']) && $_SESSION['login'] == true) {
           require_once("controller/UserController.php");
           $userController = new UserController();
                 $displayData = $userController->getData();
```

```
                $userController->Display($displayData);
            }
            break;
    case "UInstanceOp":
        require_once("controller/UserController.php");
            $userController = new UserController();
            if(isset($_REQUEST['terminateOP']) && $_REQUEST['terminateOP'] == "true")
{
                $userController->terminateInstances();
            } else {
                $userController->startInstance();
            }
        break;
    case "ATerminateOp":
        require_once("controller/AdminController.php");
        $AdminController = new AdminController();
            $AdminController->terminateInstances();
            break;
    case "default":
    default:
        require_once("controller/DefaultController.php");
        $defaultController=new DefaultController();
            if ($_GET['v']== "info") {
          $defaultController->display();
        }else {
              $defaultController->display();
            }
       break;
}
?>
</body>
</html>
```

Appendix D: Amazon Virtual Instances CPU Information

single core processor(/proc/cpuinfo)

processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5507 @ 2.27GHz
stepping : 5
cpu MHz : 2266.746
cache size : 4096 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht nx constant_tsc up nonstop_tsc aperfmperf pni ssse3 sse4_1 sse4_2 popcnt hypervisor
bogomips : 4533.49
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

4 Core 8 ECU (/proc/cpuinfo)

processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 44
model name : Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
stepping : 2
cpu MHz : 2000.068
cache size : 12288 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes

flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes hypervisor lahf_lm
bogomips : 4000.13
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 44
model name : Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
stepping : 2
cpu MHz : 2000.068
cache size : 12288 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes hypervisor lahf_lm
bogomips : 4000.13
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 2
vendor_id : GenuineIntel
cpu family : 6
model : 44
model name : Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
stepping : 2
cpu MHz : 2000.068
cache size : 12288 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes

flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes hypervisor lahf_lm
bogomips : 4000.13
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 3
vendor_id : GenuineIntel
cpu family : 6
model : 44
model name : Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
stepping : 2
cpu MHz : 2000.068
cache size : 12288 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes hypervisor lahf_lm
bogomips : 4000.13
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

8 Core 20 ECU

processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes

flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes

flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 2
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes

cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 3
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes

flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 4
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz

stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 5
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
cache size : 4096 KB
fpu : yes

fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 6
vendor_id : GenuineIntel
cpu family : 6
model : 26

model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 7
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
model : 26
model name : Intel(R) Xeon(R) CPU E5506 @ 2.13GHz
stepping : 5
cpu MHz : 2133.408
cache size : 4096 KB
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu de tsc msr pae cx8 sep cmov pat clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nonstop_tsc aperfmperf pni ssse3 cx16 sse4_1 sse4_2 popcnt hypervisor lahf_lm
bogomips : 4266.81
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

Appendix E: Glossary

AMI – Amazon Machine Image

API – Application Programming Interface

EC2 – Elastic Cloud Compute

ECU – Elastic Compute Units

IaaS – Infrastructure-as-a-Service

PaaS – Platform-as-a-Service

SaaS – Software-as-a-Service

S3 – Simple Storage Service