Final Report:
Cloud Platform Visualizing Project
Spring 2012

IRT Lab, Columbia University

Project Student
Jin Hyung Park
jp2105@columbia.edu

Mentor
Jong Yul Kim
jyk@cs.columbia.edu

Advisor
Henning Schulzrinne
hgs@cs.columbia.edu

May 7, 2012

Table of Contents

1. Introduction

1.1. Purpose

The main purpose of this project is to implement a flexible, reusable GUI client to manage and monitor various VM instances of services deployed on cloud platforms. As service monitoring on cloud platforms is still in its infancy, we believe that the prototype and code from this project will contribute to the cloud user community.

1.2. Project Scope

The project scope is defined as four goals of the project.

i. Show VM instances and status of process running within the instance

ii. Right-click menu for each instance

iii. Packaging it into an executable file

iv. Show EC2 statistics about the instances

1.3. Runtime Environment

The GUI client is written in Java. It supports Microsoft Windows 7, and Mac OS X Lion. The client is fully tested on both platforms. Currently, it supports the Amazon EC2 for the cloud platform. In addition, for this project, the visualization client is used to manage and monitor the Load Scaling Manager. Figure 1.1 shows the client.

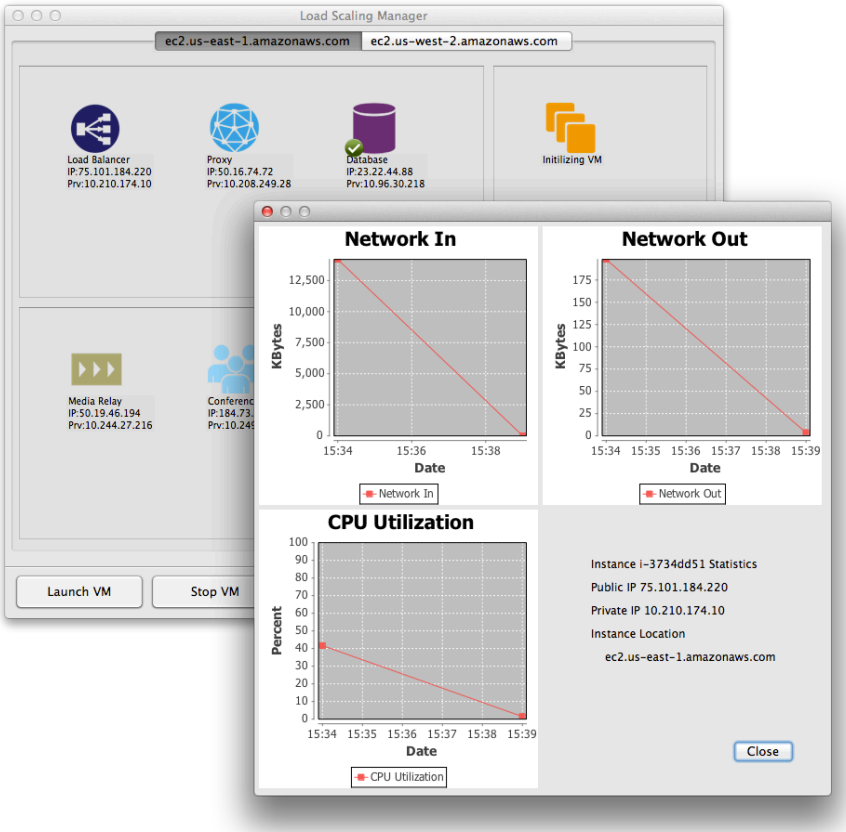

Figure 1.1 - Cloud Platform Visualizing Project

2. Application usage and features

2.1. Showing VM instances on the window

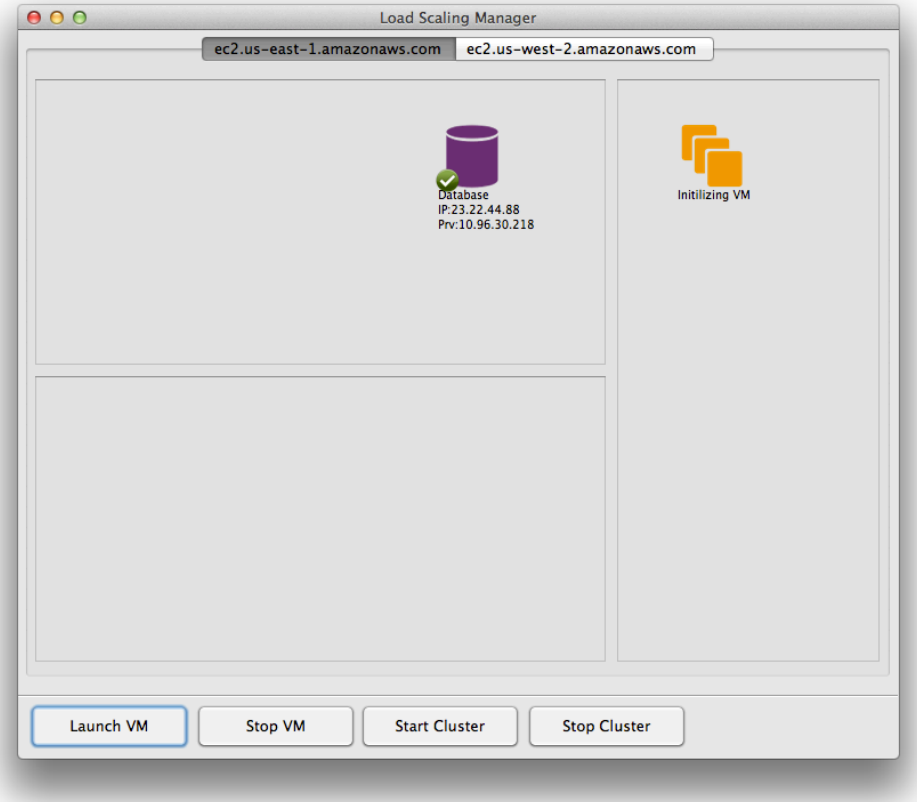

Figure 2.1 - The main window

Figure 2.1 shows the main window of our client. Currently, our client supports the Amazon EC2 cloud platform, so it contains two region tabs at the top of the window. One represents the east region, and the other represents the west region. In each region tab, we can see what VM instances are currently running. At the left upper rectangle, Load Balancer, Proxy, and Database instances will be placed. At the left lower rectangle, Media Relay, and Conference instances will be placed. At the right rectangle, waiting VM instances will be placed. Those waiting VM instances will be configured as the specific virtual machine that we need.

At the bottom of the main window, there are four buttons: “Launch VM”, “Stop VM”, “Start Cluster”, and “Stop Cluster”. “Launch VM” is used for launching the single VM instance. Specific instances can also be stopped from the right-click menu. “Start Cluster” is used for launching the pre-defined cluster of VM instances. “Stop Cluster” is used for stopping the pre-defined cluster of VM instances.

2.2. Showing the status of process running within the instance

Basically, our client shows three types of information about the VM instance on the main window. (Figure 2.2)

i. It shows the type of the VM instance. To show the kind of the VM instance, we use specific icons for each VM instance. Table 2.1 shows what icons represent what VM instances.

ii. It shows the public IP address of the VM instance. This is useful when a user tries to manage the VM instance via SSH, HTTP, and so on.

iii. It shows the private IP address. The private IP address is given by the Amazon EC2, and it is useful when we need to figure out which VM instances we are working on. This is because we cannot know its public IP address inside of its VM instance. Through the “View Status” menu, we can check the statistics of the VM instance, but this will be discussed in the later section.

Icon	Virtual Machine Type
	Empty VM
	Load Balancer
	Proxy
	Database
	Media Relay
	Conference

Table 2.1 - Each VM instance icon

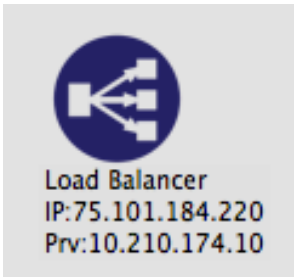

Figure 2.2 - Three types of information

In addition, each icon is defined by the VM instance tag of the Amazon EC2, so it can be easily changed if we want to use another icon for each VM instance.

2.3. Right-click pop-up menu for each instance

To manage the VM instance, our client provides the right-click pop-up menu. Figure 2.3 shows the right-click pop-up menu. The menu provides four functions: reconfiguring the VM instance, stopping the VM instance, terminating the VM instance, and viewing the

statistics information of the VM instance. However, reconfiguring the VM instance function has not yet implemented completely. This is the specific feature for the VoIP cloud platforms, I've implemented the menu holder so that it can be used later.

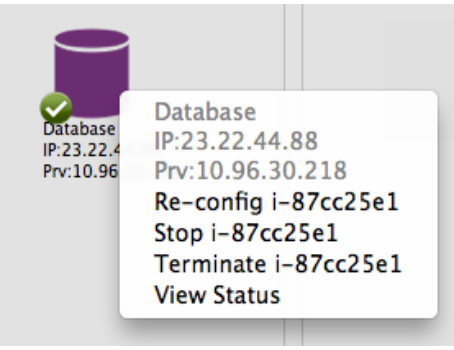

Figure 2.3 - The right-click pop-up menu

2.4. Statistics graph for each instance

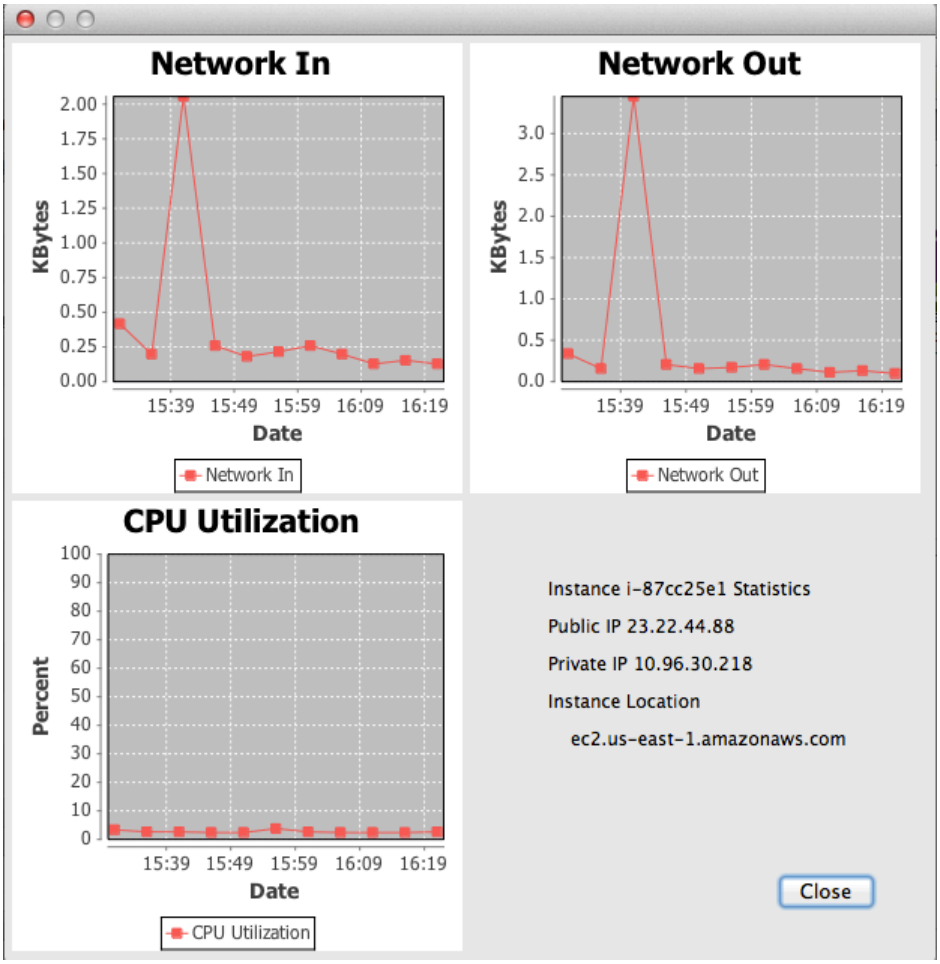

Figure 2.4 - The statistics information graph

By clicking the "View Status" on the right-click pop-up menu, we can monitor each VM instance's statistics information. There are three types of statistics information: Network Input Bytes per second, Network Output Bytes per seconds, and CPU Utilization. It shows the most recent 60 minutes of activity on the VM. With this information, we can

check the status of the VM instance. In addition, the graph is updated every 60 seconds. If the user opens up the graph window, the graph will be automatically updated in real time.

2.5. Installing the application

Our GUI client, and the Load Scaling Manager, is packaged within the DMG installer for Mac OS X. It is very easy to install. The user can install our client by just dragging and dropping the application icon. Figure 2.5 shows the mounted DMG installer. Figure 2.6 shows the installed application on Mac OS X.

Figure 2.5 - DMG Installer

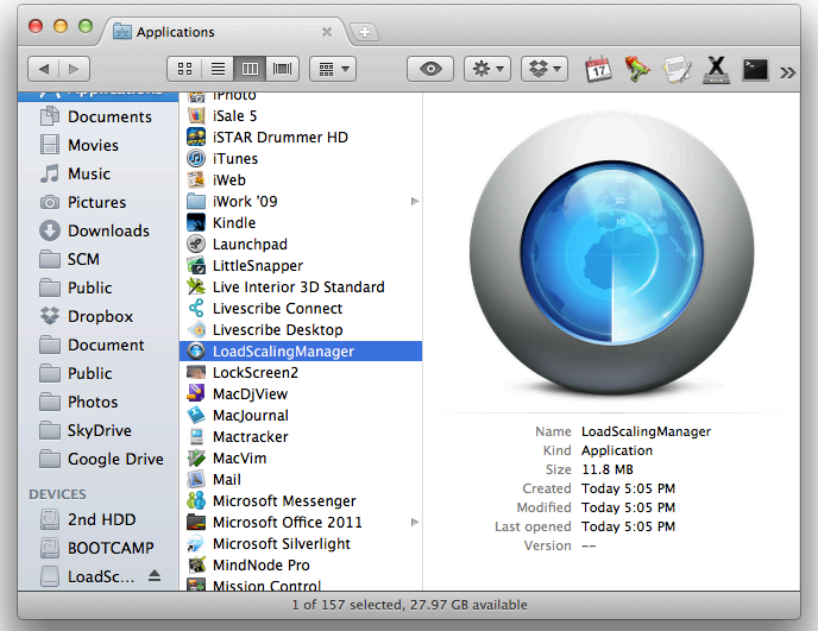

Figure 2.6 - The installed application

3. Implementation Details

3.1. Getting information on VM instances

Amazon EC2 API was heavily used in the project. The API serves as the client interface to the Amazon EC2 web service. Amazon implements the APIs using HTTP request and response under the hood.

The visualization client's UI model classes call these API functions, each in their own thread of execution, so that the user can run tasks simultaneously without being blocked by UI events and API calls.

3.2. GUI object diagram

Figure 3.1 shows us the object diagram of the visualization client.

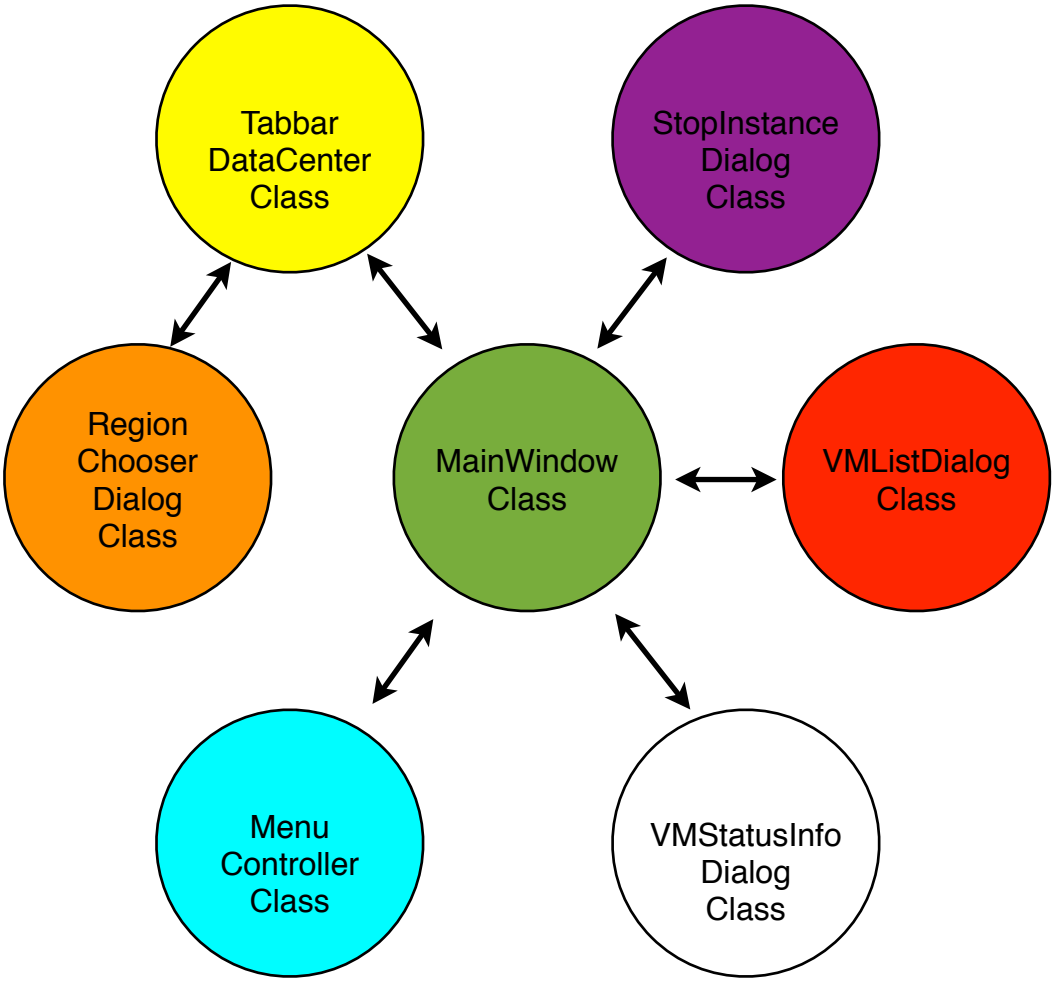

Figure 3.1 - Object diagram (Arrows mean the caller/callee of the class.)

The visualization client is written in Java with SWT framework[4], and it consists of 6 GUI classes, and one threading class: MainWindow, MenuController, RegionChooserDialog, StopInstance, TabbarDataCenter, VmListDialog, VMStatusInfoDialog class, and UIThreadWaiting class.

The MainWindow class controls the main window, which is shown when we launch the application. The Main Window class contains the Tabbar Controller, the VMList Dialog, the Pop-up menu Controller, and the VMStatusInfo Dialog.

The TabbarDataCenter controls the region group tab in the main window. With the RegionChooserDialog class, the Load Scaling Manager can determine where the user wants to launch their VM instances on. Also, through the TabbarDataCenter class, the MainWindow class switch the region displays.

The VmListDialog class manages which VM instance will be launched. When the user clicks the “Launch Instance” button on the main window, the VmListDialog class shows the VM selection dialog to the user. When the user choose the VM instance, the MainWindow class calls the Amazon EC2 API to launch the VM instance.

The StopInstanceDialog class allows the user to stop the VM instance. When the user clicks the “Stop Instance” button on the main window, the StopInstanceDialog class requests the MainWindow class to call the method that stops the process running on the VM instance.

The MenuController class controls the right-click pop-up menu on each VM instance’s icon. When the user clicks the right mouse button on the VM instance icon, the pop-up menu will be shown up. Through this menu, the user can re-configure, stop, or terminate the VM instance. Also, in this menu, the user can choose to the statistics information graph.

The VMStatusInfoDialog class shows the VM instance’s statistics information. When the user clicks the “View Status” menu, the MenuController class calls the Amazon EC2 API to get the statistics information for the VM instances, then it passes data to the VMStatusInfoDialog to display the statistics graph and other information.

Each class runs on its own thread, which are from the UIWaitingThread class. This prevents blocking the user interface.

Class Summary	
MainWindow	MainWindow class of the visualization client
MenuController	Right-click PopUp menu Controller
RegionChooserDialog	RegionChooserDialog allows users to choose the Amazon EC2 region.
StopInstanceDialog	StopInstanceDialog displays the list of VM instances so that users can stop the application running on VM instance.
TabbarDataCenter	This class represents each Amazon EC2 region.
UIWaitingThread	UIWaitingThread makes UI wait until finishing LoadScalingManager's job
VmListDialog	VmListDialog displays the list of the type of VM instances so that users can choose the VM type.
VMStatusInfoDialog	VMStatusInfoDialog shows the statistics information graph.

Table 3.1 - The summary of the package edu.columbia.cs.irt.sipcloud.gui (from JavaDoc)

3.3. Statistics graphs

VMStatusInfoDialog uses JFreeChart[5] to draw the statistics information graph. JFreeChart is a free library written entirely in Java that makes it easy for developers to display charts in their applications. JFreeChart is an open-source project, so I chose this for our project.

To draw our statistics information graph, I implemented two APIs for our project.

```
private static XYDataset createDataset(String timeName, ArrayList <Datapoint> res)
```

“createDataset” creates the XYDataset object from the Amazon EC2 results. When we request the Amazon EC2 to get the statistics information, the data are returned as “ArrayList <Datapoint>”. With this API, we can easily create the XYDataset object.

```
private static JFreeChart createChart(String timeName, String yStr, XYDataset dataset)
```

“createChart” creates the actual graph object, which is a JFreeChart object, with the given XYDataset from the “createDataset” API result. Figure 3.2 shows the JFreeChart graph result.

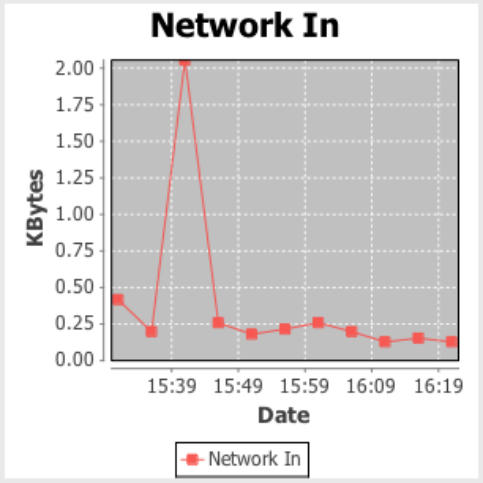

Figure 3.2 - A graph drawn using JFreeChart

3.4. Application packaging

As I mentioned in the Section 2.5, the visualization client and the Load Scaling Manager is packaged for Mac OS X. I used one Bash shell script, and the tool iDMG to create the DMG Installer for the Mac OS X. In the project source file, there is the Bash shell script, which is named “packageMac.sh” in the project root directory. This “packageMac.sh” shell script creates the Mac OS X application bundle. Before running this script, we need to export our compiled project to one JAR file in the Eclipse.

4. Lessons learned

4.1. Desktop GUI Client Programming in Java

In this project, I used Java and the SWT framework to implement a desktop GUI program. That was my first time to use them for GUI programming. This was helpful in that I acquired the skills to implement desktop GUI applications without the platform dependency.

4.2. Lessons from the Amazon EC2

First, as the main part of this project, I implemented managing and monitoring the VM instance. By working on the project, I found out that the best benefit of cloud computing is that we can spawn and use a virtual machine whenever we want. We don't have to run the machines all the time but only when we need to. Additionally, we can scale the number of machines easily by simply clicking or calling a function in Amazon EC2 API. The best benefit is that we do not need to run always all machines. By using Amazon EC2 APIs, we can turn on or off each virtual machine whenever we want. Additionally, we can add the scalability easily. With just one click, we can add or delete the machine at once.

Second, it was a really valuable experience for me to play around with EC2 APIs. It is hard for students to get this kind of experience. Exposure to CloudWatch monitoring API was especially valuable and I'm certain that it will be helpful whenever I have to work on another project related to cloud computing.

5. Future Work Ideas

5.1. Supporting other cloud platforms

At this moment, our GUI client supports only the Amazon EC2 cloud platform. To use more generally, we can consider supporting other cloud platforms like Google App Engine, or Microsoft Azure Cloud Platform.

5.2. Packaging for Microsoft Windows

Currently, the visualization client comes with a packaging script for Mac OS X; however, it would be useful if it can be installed via the Windows Installer even though our current application can run without any installation tool.

Appendix A: References

[1] Amazon EC2 - http://aws.amazon.com/ec2/
[2] Google App Engine - https://developers.google.com/appengine/
[3] Microsoft Azure Platform - http://www.windowsazure.com/en-us/
[4] SWT Framework - http://www.eclipse.org/swt/
[5] JFreeChart - http://www.jfree.org/jfreechart/