**1.0 ABSTRACT**

Cloud Computing is the hottest trend in the IT World and each and every company is slowly but surely getting engrossed and migrating to this big thing. In this project report we have explained the basic concepts related to Cloud Computing and its types. Further we have explored private cloud in detail by providing Eucalyptus as an example. To utilize its benefits and to observe its working, we have setup a private cloud in our laboratory connecting all the terminals following the High Availability (HA) architecture of the Eucalyptus Platform. By exploring its architecture and functioning in detail, we observed a lack of interface for the management of private cloud and thus we will make a cloud management platform that manages the cloud in the best possible manner considering client’s experience.

Our GUI makes use of boto python modules which are python commands supported by eucalyptus. Boto is a python interface to eucalyptus cloud services. Each boto command represents some functionality by running various processes simultaneously. Since euca-commands cannot be directly executed on the browser, we make use of php scripting language as a medium to connect euca-commands with HTML.

The integrated performance monitoring service in our GUI provides with metric parameters of the instances giving the user an idea of the network and CPU load of the cloud controller. We have tried and test the various linux commands that give information about the performance monitoring.

**2.1** **REVIEW OF LITERATURE**

Traditional dedicated servers also called as on-premise servers are the ones which are situated inside the standalone data center. They require power, cooling, networks, bandwidth and infinite amount of storage. In addition to above, it cannot work without a software stack and a team of experts required for starting , configuring and running the system .They are also responsible for product development, staging, testing and also for failures in various environments. On top of it any new version or update results in the destruction of the whole software stack. Because of all these factors, business applications deployed on this infrastructure are very expensive and thus the small business companies can’t just afford it. On the contrary cloud computing is much faster as it is just plug n play and the platform is ready. It requires no technical experts, no severs , no storage and no upgrades but just directly customizing applications which are on the shared data center based on the client’s needs. Since the burden of initial setup is eliminated the company is up and running in few days instead of hard long lasting months for its setup.

National Institute of Standard and technology ( NIST) defines cloud computing as “a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” (Mell & Grance, 2009, p. 1).The NIST definition of cloud computing includes five essential characteristics (on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service), three service models (infrastructure, platform, software), and three deployment models (private, hybrid, public) .

*Pillars of Cloud Computing*

*Orchestration*

Orchestration describes the automated arrangement, coordination, and management of complex computer systems, middleware, and services. Orchestration also provides centralized management of the resource pool, including billing, metering, and chargeback for consumption.

*Virtualization*

Virtualization is a method of running multiple independent virtual operating systems on a single physical computer.  It is a way of maximizing physical resources to minimize the investment in hardware.

*Multitenancy*

Multitenancy refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple client organizations (tenants). With a multitenant architecture, a software application is designed to virtually partition its data and configuration, and each client organization works with a customized virtual application instance.

*Why Cloud Computing?*

-Low setup cost (pay for only what you use)

The billing is on hourly or monthly basis. You pay only for the resources your actually consume. This is unlike the traditional services where you pay a fixed amount even if you don’t use the resources, or don’t have enough clients to consume the preconfigured resources.

-More Scalable

Cloud is elastic in nature, i.e., you can control the number of resources you use at any given point in time. Compare this with traditional hosting, where you rent a fixed number of resources for fixed amount of time. Based on your computing requirements and configuration, your cloud service provider can respond quickly to scale *up or* down i.e. Auto *Scaling.*

-More Reliable

Most people seem to follow the idea that if you have a Dedicated Server in a datacenter and it fails someone will immediately run to it and fix it but it won’t be an instant fix, this is because someone’s actually got to fix the problem and this takes time. The difference with the Cloud is that its providers have many data centre’s located all over the world, so if any instance goes down, or even an entire datacenter goes down, you just start up a new instance somewhere else, the problem is solved in a matter of minutes.

-Regular Upgrading

Cloud Vendors regularly upgrade their software, so that the users don’t have to put any effort into installing and upgrading the applications. This enhances the user experience and thus its utility.

-Ubiquitous Access

A major advantage of cloud is that it can easily and quickly be accessed from anywhere with a web browser. This gives users a great facility even when they are at home or in another country. They can access real time synchronized applications from Laptops and Smart Phones.

*GENERIC ARCHITECTURE*

When talking about a cloud computing system, it's helpful to divide it into two sections: the **front end** and the **back end**. They connect to each other through a network, usually the Internet. The front end is the side the computer user, or client, sees. The back end is the "cloud" section of the system.

The front end includes the client's computer (or computer network) and the application required to access the cloud computing system. Not all cloud computing systems have the same user interface.

On the back end of the system are the various computers, servers and data storage systems that create the "cloud" of computing services. Usually, each application will have its own dedicated server.

A central server administers the system, monitoring traffic and client demands to ensure everything runs smoothly. It follows a set of rules called **protocols** and uses a special kind of software called **middleware**. Middleware allows networked computers to communicate with each other.

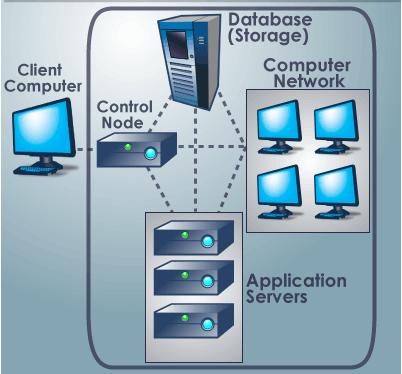


Fig.1 Cloud Computing Architecture

The above given Figure 1 gives the general understanding of the organization of the different components of cloud computing.

|  |  |  |  |
| --- | --- | --- | --- |
| Characteristic | Public Cloud | Private Cloud | Hybrid Cloud |
| Scalability | Very High | Limited | Very High |
| Security | Good, but depends on the security measures of the service provider. | Most secure, as all storage is on-premises. | Very secure, integration options add an additional layer of security. |
| Performance | Low to Medium | Very Good | Good, as active content is cached on-premises. |
| Reliability | Medium, depends on the Internet connectivity and service provider availability | High, as all equipment is on-premises | Medium to High, as cached content is kept on-premises, but also depends on connectivity and service provider availability |
| Cost | Very good, pay-as-you-go model and no need for on-premises storage infrastructure | Good, but requires on-premises, such as data center space, electricity and cooling | Improved, since it allows moving some of storage resources to a pay-as-you-go model |
| Example | Amazon Elastic Compute Cloud (EC2), Google AppEngine etc. | [Ubuntu Enterprise Cloud](http://www.ubuntu.com/cloud/private)(Eucalyptus), [Microsoft Azure](http://www.microsoft.com/windowsazure) etc. | --- |

Table 1: Comparison between different types of cloud

As private cloud is built on the existing on-premise infrastructure, migrating from traditional server configuration to cloud computing architecture is not prohibited, thus it is very flexible. Since cloud computing architecture depends on the way it has been developed, it varies from developer to developer. Hence each developer can customize the cloud architecture according to its needs. Even as security is the prime factor for judgment, private cloud is more superior to public cloud as in this case the user doesn’t have to give the private information to its cloud service provider. Thus in this paper we have explored eucalyptus private cloud which is a software platform for on-premise (private) clouds. It is offered by Eucalyptus Systems, which provides IT organizations and technology businesses with this platform for on-premises private clouds.

Eucalyptus

Eucalyptus was originally built as an open source cloud product and now supports enterprise-class private cloud as well as hybrid cloud computing. Eucalyptus uses existing infrastructure to create a scalable, secure web services layer that abstracts compute, network, security groups and storage to offer IaaS.

*Eucalyptus Components*

Eucalyptus is comprised of six components: Cloud Controller (CLC), Walrus, Cluster Controller (CC), Storage Controller (SC), Node Controller (NC) and an optional VMware Broker (Broker or VB). Other than the VMware Broker, each component is a stand-alone web service. This architecture allows Eucalyptus both to expose each web service as a well-defined, language-agnostic API, and to support existing web service standards for secure communication between its components.

Detailed description of each Eucalyptus component follows:

*Cloud Controller*

The Cloud Controller (CLC) is the entry-point into the cloud for administrators, developers, project managers, and end-users. The CLC queries other components for information about resources, makes high-level scheduling decisions, and makes requests to the Cluster Controllers (CCs). As the interface to the management platform, the CLC is responsible for exposing and managing the underlying virtualized resources (servers, network, and storage). You can access the CLC through command line tools that are compatible with Amazon’s Elastic Compute Cloud (EC2) and through a web-based Dashboard.

*Walrus*

Walrus allows users to store persistent data, organized as buckets and objects. You can use Walrus to create, delete, and list buckets, or to put, get, and delete objects, or to set access control policies. Walrus is interface compatible with Amazon’s Simple Storage Service (S3), providing a mechanism for storing and accessing virtual machine images and user data. Walrus can be accessed by end-users, whether the user is running a client from outside the cloud or from a virtual machine instance running inside the cloud.

*Cluster Controller*

The Cluster Controller (CC) generally executes on a machine that has network connectivity to both the machines running the Node Controllers (NCs) and to the machine running the CLC. CCs gather information about a set of NCs and schedules virtual machine (VM) execution on specific NCs. The CC also manages the virtual machine networks. All NCs associated with a single CC must be in the same subnet.

*Storage Controller*

The Storage Controller (SC) provides functionality similar to the Amazon Elastic Block Store (Amazon EBS). The SC is capable of interfacing with various storage systems (NFS, iSCSI, SAN devices, etc.). Elastic block storage exports storage volumes that can be attached by a VM and mounted or accessed as a raw block device. EBS volumes persist past VM termination and are commonly used to store persistent data. An EBS volume cannot be shared between VMs and can only be accessed within the same availability zone in which the VM is running. Users can create snapshots from EBS volumes. Snapshots are stored in Walrus and made available across availability zones. Eucalyptus with SAN support lets you use your enterprise-grade SAN devices to host EBS storage within a Eucalyptus cloud.

*Node Controller*

The Node Controller (NC) executes on any machine that hosts VM instances. The NC controls VM activities, including the execution, inspection, and termination of VM instances. It also fetches and maintains a local cache of instance images, and it queries and controls the system software (host OS and the hypervisor) in response to queries and control requests from the CC. The NC is also responsible for the management of the virtual network endpoint.

*VMware Broker*

VMware Broker (Broker or VB) is an optional Eucalyptus component, which is available if you are a Eucalyptus subscriber. VMware Broker enables Eucalyptus to deploy virtual machines (VMs) on VMware infrastructure elements. VMware Broker mediates all interactions between the CC and VMware hypervisors (ESX/ESXi) either directly or through VMware vCenter.

After understanding the basic functionalities of each component, we will now unearth the Eucalyptus architecture.

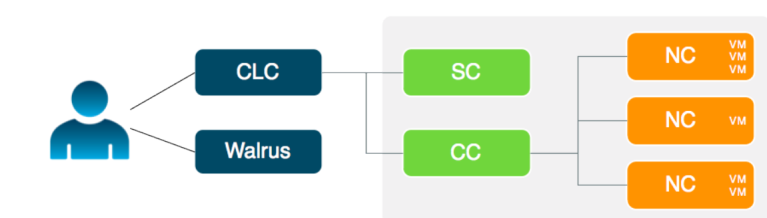


Fig 2.Eucalyptus Architecture

*Working*

The cloud components, Cloud Controller (CLC) and Walrus, communicate with cluster components, the Cluster Controllers (CCs) and Storage Controllers (SCs). The CCs and SCs, in turn, communicate with the Node Controllers (NCs). The networks between machines hosting these components must be able to allow TCP connections between them. However, if the CCs are on separate network interfaces (one for the network on which the cloud components are hosted and another for the network that NCs use) the CCs will act as software routers between these networks in some networking configurations. So each cluster can use an internal private network for its NCs and the CCs will route traffic from that network to a network shared by the cloud components. Virtual machines (VMs) run on the machines that host NCs. You can use the CCs as software routers for traffic between clients outside Eucalyptus and VMs. Or the VMs can use the routing framework already in place without CC software routers. However, depending on the layer-2 isolation characteristics of your existing network, you might not be able to implement all of the security features supported by Eucalyptus.

**2.2 EXISTING SYSTEM**

**RightScale Management Platform**

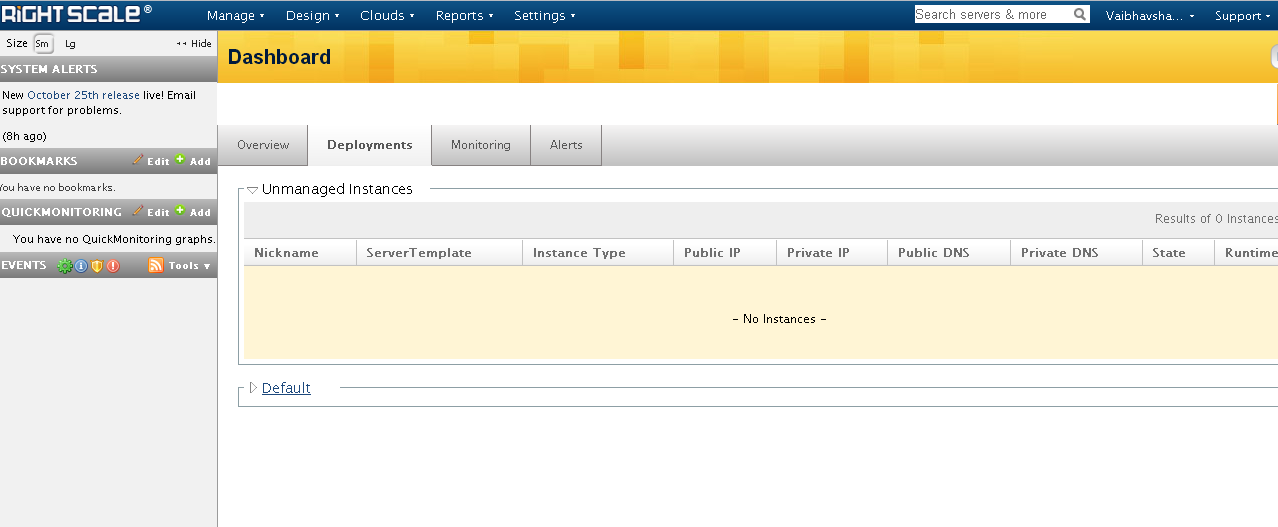


Fig3. RightScale Platform

RightScale enables the client to manage his entire cloud infrastructure with a single, integrated solution. It provides **abstraction with complete customization**, allowing you to focus on your applications while optimizing your infrastructure where necessary. In short, RightScale makes cloud slightly manageable. **Automation is the core** of RightScale, freeing you to run efficient, scalable, and highly-available applications. And rest assured knowing that you have **visibility and control** over all of client’s resources in one place. To take advantage of best practices, it encourages client to **tap into cloud expertise** provided by our service, support, and partner networks when building and managing your infrastructure.

**Drawbacks in this system:**

1. Too Expensive

For a Month Subscription it costs a huge sum of around $1000.Thus small companies can’t afford it.

2. Highly Complex

The RightScale GUI is complex as it uses cloud computing jargon thus it makes difficult for the client to operate and use the functionalities.

3. Lack of Performance Monitoring tool

RightScale GUI does not take into account the availability of resources at any point in time and thus the waiting time of the client increases leading to his dissatisfaction.

**3.0 PROBLEM DEFINITION**

One of the major shortcomings of using Eucalyptus private Cloud is that it lacks an user interface for its management and deployment of instances. Because of this every client is fully dependent on the administrator to carry out most of the operations such as creating an instance, terminating an instance, creating Volume and taking Snapshots. In addition to this even the administrator has to run scripts to check and monitor the instances which take a lot of time. Even the client faces the problem of slow processing due to lack of hardware resources during peak load and thus he is not able to carry out the required work smoothly.

To realize the opportunities of the cloud, you need cloud management that gives you the extreme efficiency, speed, and control to navigate and seize these opportunities. To overcome these shortcomings we propose to make a Cloud Management Platform which will consist of an easily usable Graphical User Interface allowing the clients to operate independently. The User Interface will allow the client to start an instance, to terminate an instance, assign a volume to it and will make a snapshot of it. It will also provide additional features such as adding users, giving priorities to users and accordingly assigning key pairs to them. It will also incorporate performance monitoring feature that will help the admin with warnings and alerts as and when there is shortage of resources so that he can handle the situation in the right manner. In short, this cloud management platform will be useful to both the client as well as the admin.

**4.0. PROPOSED SYSTEM**

**Cloud Management** Platform provides a transparent "single pane of glass" view into your entire cloud infrastructure. The client can conveniently access his private cloud and resource pools from one Dashboard. Here the client can provision entire server deployments in minutes and then automate and govern them over their lifetimes. Everything is visible, organized, and controlled. And it works across various private clouds.

With our cloud management platform, managing 10 or 1,000 servers can be equally simple. **Our Cloud Management Platform** bridges the gap between your applications and your cloud infrastructure. The [**Automation Engine**](http://www.rightscale.com/products/automation-engine.php) feature gives you the power to provision, monitor, scale, and manage entire server deployments efficiently and reliably. Automation is what makes the cloud's scalability and low cost possible. But if automation isn't done properly it will actually result in more work for IT staff and thus would hamper their growth. Thus we have designed and developed a Graphical User Interface for managing the private cloud as shown in the figure given below.

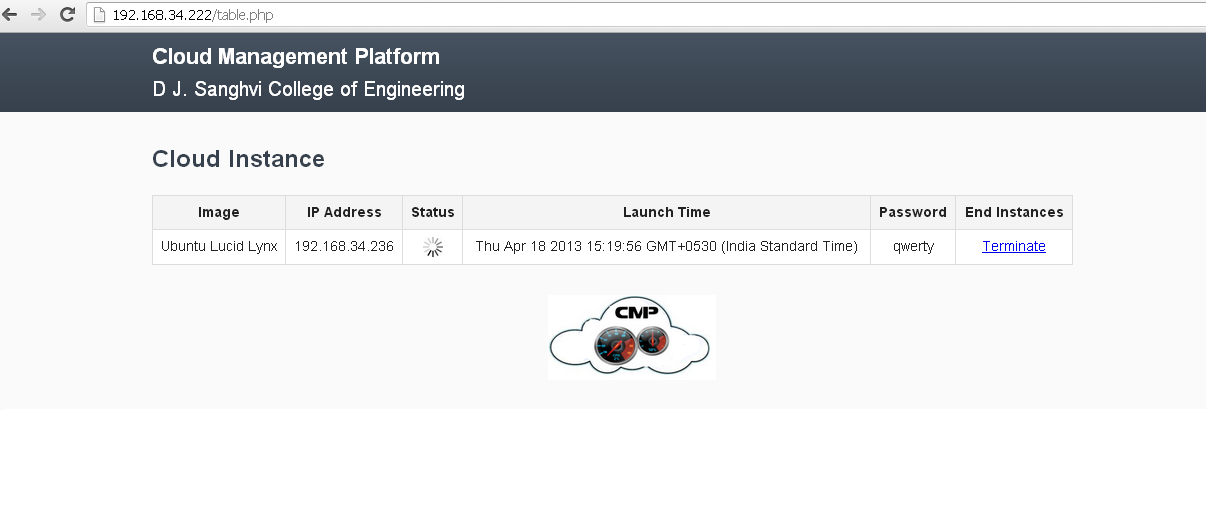


Fig 4.Graphical User Interface

The various options given in the figure are explained in detail in the following section.

1. Image

Provides the image name of the emi which is set for execution.

2. IP address

Public IP address assigned to the instance.

3. Status

Provides the status of the instance. ( Pending / Running / Shutting-Down / Terminated).

4. Launch Time

Time of launch of the instance.

5. Password

Displays the pass code required when instances are viewed using TightVNC.

6. End Instances

Terminates the instances.

**4.1 FEASIBLITY STUDY OF PROPOSED SYSTEM**

This being a Cloud Computing project, feasibility of all sections have to be considered and it is of utmost importance.

**4.1.1 Economic Feasibility:**

Economics of this project need to be understood from various perspectives. Major perspectives would be the availability of the hardware requirements for cloud setup. And these requirements include Compute, Storage and Network Requirements which are explained as follows:

Compute Requirements

• Physical Machines: All Eucalyptus components must be installed on physical machines, not virtual machines.

• Central Processing Units (CPUs): It is recommended that each machine in your Eucalyptus cloud contain either an Intel or AMD processor with a minimum of two, 2GHz cores.

• Operating Systems: Eucalyptus supports the following Linux distributions: CentOS 5 (5.6 and above), CentOS 6, RHEL 5 (5.6 above), RHEL 6, Ubuntu 10.04 LTS, Ubuntu 11.04 Natty Version, and Ubuntu 12.04 LTS.

Storage and Memory Requirements

• Each machine in your network needs a minimum of 30 GB of storage.

• We recommend at least 100GB for Walrus and SC hosts running Linux VMs. It is recommended at least 250GB for Walrus and SC hosts running Windows VMs.

Network Requirements

• All NCs must have access to a minimum of 1Gb Ethernet network connectivity.

• All Eucalyptus components must have at least one Network Interface Card (NIC) for a base-line deployment.

Apart from these requirements, we also require a public IP for cloud and Internet access to make our GUI ubiquitously accessible.

**4.1.2 Technical feasibility:**

The main technical issue is regarding the constant uptime of the cloud controller for the working of the entire cloud. To maintain the cloud, a person of expert knowledge in cloud computing domain is required. During the installation of cloud, many technical constraints are faced such as getting Internet access, Linux scripting etc.

**4.1.3 Operational Feasibility:**

Clientele today wants access on the go, no dependence on the admin, no waiting time and very user friendly platform, hence our cloud management platform will be an ideal solution for the clients to make optimum use of cloud computing.

|  |  |
| --- | --- |
| WHAT THE USER WANTS,  WHAT THE USER NEEDS | CMP REQUIREMENT |
| Access on the go | As our GUI is Internet based, the client can access it whenever he feels the need of it through any of the current generation gadgets. |
| No dependence on the admin. | Since our GUI will eliminate the need of scripting, the client would no longer be dependent on the admin to perform any required functions. |
| User friendly | Simple and sophisticated GUI makes the client’s work easy and quick. |
| No waiting time | Performance monitoring feature will dynamic allocate the resources and thus will decrease the “slow time” eventually enhancing the performance.. |

Table 2: Features requirements

**5.0 PROJECT MANAGEMENT**

**5.1 ESTIMATION AND PLANNING**

**5.1.1 Function Point Calculation:**

*I Client Inputs:*

a) User Identification

b) IP address

c) Instance id

d) Security key

*II Client outputs:*

a) Instance ID

b) Performance of the Instance

c) Storage Space

*III No. of Client Enquiries*

a) Instances

b) Images

c) Key Pair

d) Elastic IPs

e) Security Groups

f) Volumes and Snapshots.

g) Performance Monitoring.

*IV No. of files*

a) Transaction log file.

b) Daily report.

c) User Registration.

d) Warning File

e) Performance File

f) Instance ID File

*V. No of admin enquiries.*

a) Daily, weekly, monthly reports.

b) View Hardware Resources.

c) View Performance.

d) View feedback.

e) Action required on Downtime

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. no | Domain Characteristics | Count | \* | Average Value |  |
| 1 | No. of Client Inputs | 4 | \* | 4 | 16 |
| 2 | No. of Client Outputs | 3 | \* | 5 | 15 |
| 3 | No. of Client Enquiries | 7 | \* | 4 | 28 |
| 4 | No. of Files | 6 | \* | 10 | 60 |
| 5 | No. of Admin Enquiries | 5 | \* | 7 | 35 |
|  |  |  |  |  | 154 |

Table 3: FP Calculation

Now the software complexity can be computed by answering following questions. These are complexity adjustment values.

1. Does the system need reliable backup and recovery?

2. Are data communications required?

3. Are distributed processing systems used?

4. Is performance of system critical?

5. Will the system run in an existing operational environment?

6. Does the system require online data entry?

7. Does online data entry require input transaction to be built over multiple screens or operations?

8. Are master files updated online?

9. Is the internal processing complex?

10. Are the inputs, outputs, files or enquiries complex?

11. Is the code reusable?

12. Are conversion and installation included in design?

13. Is the system designed for multiple installations in different organizations?

14. Is the application designed to facilitate change and ease of use by user?

Let us assume that the complexity adjustment values are average, i.e 4.

14

∑ Fi = 14 x 4 = 56

i=1

FP =Count Total x (0.65 + 0.01 x 56)

= 154 x (1.21)

= 186.34

Therefore, **FP = 186.34**

**5.1.2 COCOMO (Constructive COst MOdel):**

We will be using basic COCOMO as the cost estimation model for quick, early and rough order of magnitude estimates of software costs. COCOMO uses 3 levels of project complexity:

1. Organic project

2. Semi-detached project

3. Embedded project

can be classified as a semi-detached mode project.

The equations in this model are:

E = ab(KLOC)bb

D = Cb(E)db

P = E / D

Where E is the effort applied in person-months,

D is the development time in chronological months.

KLOC is the kilo-lines of code

The coefficients ab, bb, cb, db for semi detached mode are as follows:

ab = 3.0

bb = 1.12

cb = 2.5

db = 0.35

Now, we assume that the estimated kilo-lines of code in are 10KLOC.

Effort Estimation:

E = 3.0 x (10)1.12

= 39.55

= 40 person-months approx.

Duration estimation:

D = 2.5 x (40)0.35

= 9.09

= 9 months approx.

Persons Estimation:

P = 40 / 9

= 4.44

= 4 persons approx.

**5.1.3 Project Work Breakdown Structure**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start Date** | **End Date** | **Duration(days)** |
|  |  |  |  |
| Project Topic Selection |  |  |  |
| 1.Discussing different topics | 1-8-2012 | 7-8-2012 | 6 |
| 2.Finalizing Project | 7-8-2012 | 8-8-2012 | 1 |
|  |  |  |  |
| Research Work |  |  |  |
| 1.Performed Literature Survey | 8-8-2012 | 14-8-2012 | 6 |
| 2.Researched on Cloud Computing | 14-8-2012 | 21-8-2012 | 6 |
| 3.Formulated the platform to be used | 21-8-2012 | 30-8-2012 | 9 |
| 4.Future Scope(Data mining technique) | 30-8-2012 | 5-9-2012 | 6 |
| 5.Learning basic Linux Scripting | 24-9-2012 | 9-10-2012 | 15 |
| 6.Exploring the existing systems | 5-9-2012 | 11-9-2012 | 6 |
|  |  |  |  |
| Planning and Framework |  |  |  |
| 1.Designed the GUI of Proposed System | 11-9-2012 | 17-9-2012 | 6 |
| 2.Discussed the architecture to be used | 17-9-2012 | 26-9-2012 | 9 |
| 3.Installing the private cloud | 24-8-2012 | 24-9-2012 | 30 |
| 4.Researched on Performance Monitoring | 27-9-2012 | 29-9-2012 | 2 |
| 5.Designed flowcharts | 29-9-2012 | 30-9-2012 | 1 |
| 6.D  ocumented a paper for Springer | 1-10-2012 | 20-10-2012 | 20 |
|  |  |  |  |
| Final Report + Presentation |  |  |  |
| 1.Making Report | 20-10-2012 | 26-10-2012 | 6 |
| 2.Making Presentation | 26-10-2012 | 28-10-2012 | 2 |
| 3.Report Submission | 28-10-2012 | 29-10-2012 | 1 |
| 4.Presentation Day | 29-10-2012 | 30-10-2012 | 1 |
|  |  |  |  |
| Implementing Private Cloud |  |  |  |
|  |  |  |  |
| Installing the Cloud Controller and Node Controller | 14-1-13 | 17-1-13 | 4 |
| Bundling Instances | 18-1-13 | 21-1-13 | 3 |
| Making Security Key Groups and KeyPairs | 22-1-13 | 28-1-13 | 6 |
|  |  |  |  |
| Implementing GUI and Scripts |  |  |  |
|  |  |  |  |
| Designing the GUI | 29-1-13 | 4-2-13 | 6 |
| Enhancing the performance tool | 5-2-13 | 8-2-13 | 4 |
| Making Desired Scripts | 9-2-13 | 20-2-13 | 10 |
|  |  |  |  |
| Integrating and Testing |  |  |  |
|  |  |  |  |
| Integrating Private Cloud an GUI | 21-2-13 | 26-2-13 | 5 |
| Development of Test Cases | 27-2-13 | 5-3-13 | 6 |
| Unit & Integration Testing | 6-3-13 | 19-3-13 | 12 |
|  |  |  |  |
| Running & Analyzing |  |  |  |
|  |  |  |  |
| Run the written Scripts | 20-3-13 | 23-3-13 | 4 |
| Analyzing the system performance with modifications | 24-3-13 | 1-4-13 | 7 |
|  |  |  |  |
| Final Report |  |  |  |
|  |  |  |  |
| Making Black book | 2-4-13 | 10-4-13 | 8 |
| Making Presentation | 11-4-13 | 16-4-13 | 5 |

Table 4: Work Breakdown

**5.1.4 Gantt Chart**

****

Fig. 5 Gantt Chart

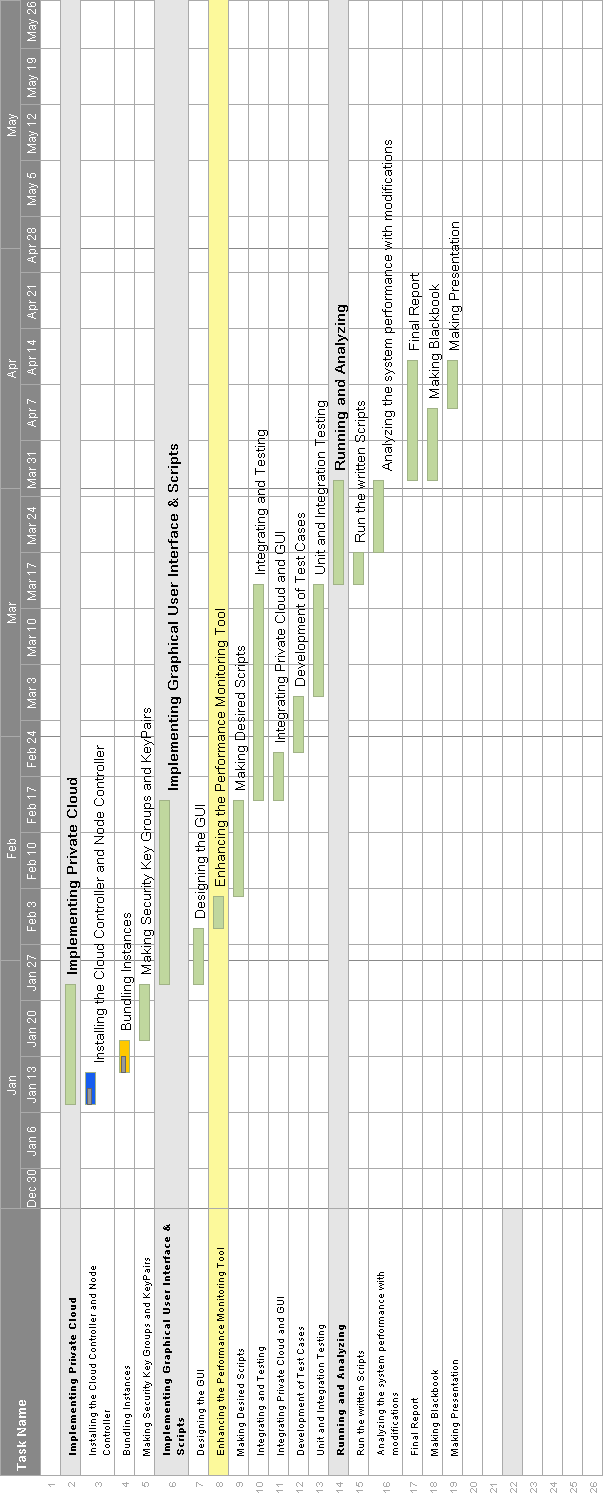


Fig.6 Gantt Chart

**5.2 PROJECT RESOURCES**

**5.2.1 Hardware Requirements**

-Physical virtually enabled machines with minimum two cores of 2 GHz.

-Each machine should have 100GB storage.

-Every machine should have a NIC (Network Interface Card).

- All NCs must have access to a minimum of 1Gb Ethernet network connectivity.

**5.2.2 Software Requirements**

- Eucalyptus supports the following Linux distributions: CentOS 5 (5.6 and above), CentOS 6,

RHEL 5 (5.6 above), RHEL 6, Ubuntu 10.04 LTS, Ubuntu 11.04 Natty Version, and Ubuntu 12.04 LTS

-Browser Compatibility

-Public IP

**5.2.3 Operating Environment**

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been ported to more computer hardware platforms than any other operating system. It is a leading operating system on servers and other big iron systems such as mainframe computers and supercomputers more than 90% of today's 500 fastest supercomputers run some variant of Linux, including the 10 fastest. Linux also runs on embedded systems (devices where the operating system is typically built into the firmware and highly tailored to the system) such as mobile phones, tablet computers, network routers, televisions and video game consoles; the Android system in wide use on mobile devices is built on the Linux kernel.

Linux distributions have taken a wide variety of forms—from fully featured desktop, server, laptop, netbook, mobile phone, and tablet operating systems as well as minimal environments (typically for use in embedded systems or for booting from a floppy disk). Aside from certain custom software (such as installers and configuration tools), a distribution is most simply described as a particular assortment of applications installed on top of a set of libraries married with a version of the kernel, such that its "out-of-the-box" capabilities meet most of the needs of its particular end-user base.

One can distinguish between commercially backed distributions, such as Fedora (Red Hat), [openSUSE](http://en.wikipedia.org/wiki/OpenSUSE" \o "OpenSUSE) (SUSE), Ubuntu (Canonical Ltd.), and Mandriva Linux ([Mandriva](http://en.wikipedia.org/wiki/Mandriva" \o "Mandriva)), and entirely community-driven distributions, such as [Debian](http://en.wikipedia.org/wiki/Debian" \o "Debian), [Mageia](http://en.wikipedia.org/wiki/Mageia" \o "Mageia) and [Gentoo](http://en.wikipedia.org/wiki/Gentoo_Linux).

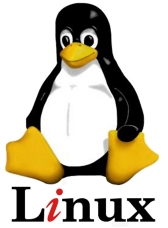


Fig.7 Linux Mascot

Since Eucalyptus platform comes with Linux flavors only, we are using Linux Server to setup our cloud but the client can bundled any operating system instances.

**6.0 PROJECT DESIGN**

**6.1 Overall Description**

**6.1.1 Project Scope**

Since the GUI frees the client-admin coordination required to take necessary actions, it’s a plus point to both admin as well as the client. The GUI attracts the clients as it removes the over-head complexity required in utilizing the features of cloud and thus enhances the utility of cloud. The traditional servers can be shifted to private cloud without many changes and thus small and midscale companies after installing the GUI will not shy away from exploring the advantages of private cloud. To add to it, the performance monitoring tab will avoid overloading of client’s instances and also support the admin in accelerating his work with the help of guidelines and alerts. The project scope is not restricted just to Eucalyptus private cloud but it can be extended even to other private clouds such as Open Nebula and Nimbus.

As already explained, private cloud is a next big thing in IT World because of all the advantages and the flexibility it offers. Thus it generates the necessity for having a cloud management platform that can efficiently manage the cloud and even enhance the user experience since it shields the user from the complicated private cloud architecture and also from the performance monitoring . The above mentioned things have been incorporated in our GUI, thus making it an ideal cloud management platform. Regarding performance monitoring which is a great helping option to the client, we will make a database of all the proposed parameters and will check each of their values in each and every possible scenario to explore each and every possibility after installing the cloud in our private network. For extreme efficiency and accuracy, we would give these values of database to the associative rule data mining technique which would assess each of the cases and would give the most optimum result or solution in case of some problem. In case any scenario exceeds the threshold decided by the data mining technique it will suggest or throw a warning to the admin to take the required actions to solve it.

**6.1.2 Project Purpose**

Currently cloud computing is the most trending topic but its benefits cannot be utilized to its full extent as the focus is on the functionalities and not on the clients which actually use them. As stated above there are existing system which does the same work but in a complex manner giving an unpleasant user experience. Thus the client doesn’t feel being part of the whole process even after paying huge chunks of money and hence doesn’t use it in the best possible manner. Considering all the factors, we felt that a pleasant, user friendly, less complex Graphical User Interface would be an apt solution for the clients as it would reduce the efforts required and thus accelerating the entire process.

**6.1.3 Project Features**

The major project features are aimed at addressing the major concerns like: Simplicity, Cost effectiveness, User friendly and Performance Monitoring.

1. Simple & Sober – All the functions to be done by the client is done by simple mouse clicks.

2. Time redemptive – Since our GUI eliminates the need of admin-dependence, all the functions of client are on-click and hence time is saved.

3. Minimized Human Intervention - Automatic processing of client’s request.

4.Performance Monitoring-This tab will help in allocating the resources considering each and every scenario of the proposed parameters and thus reducing the slow time of important processes.

**6.1.4. Design Issues and Constraints**

1. Virtually Enabled terminals are a necessity for our project.

2. Entire management is done using the platform; therefore it has to be working 24\*7.

3. On-demand scripting is not supported in our GUI.

**6.2 Information Description**

**6.2.1 Data Flow Diagram**

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to , and where the data will be stored. It does not show information about the timing of processes , or information about whether processes wll opearate in sequence or in parallel.

Cloud request

Node Controller

Cloud Controller

Client

Sends information

Instance ID

Access Granted

Fig.8 LEVEL 0 DFD

In the above shown level 0 DFD : the Client will provide its Instance ID and will request to the Cloud Controller(CC) for access. CC after authenticating the Client will grant him the access. The CC will even send the required information to the Node Controller (NC) for creating the Instance.

Cloud Request

Sends Information

Instance ID

Node Controller

Cloud Controller

Client

Access Granted

OS Requests

Sends Information

List of OS

Cloud Controller

Fig.9 LEVEL 1 DFD

The above diagram is the level 1 DFD. The main difference with respect to the level 0 DFD is that, here the Client will be able to search for the required Operating System (OS) from the list of available OSs. And the other connections are well shown above.

**6.3 System Design**

**6.3.1 System Architecture Design**

Node Controller

Cloud Controller

Connects Cloud Using

Graphical User Interface

Instances

Fig.10 System Architecture

In this architecture, our GUI will connect to the cloud controller whenever the instance tries to use it for any of the functionalities.Then the cloud controller will check the authenticate client’s instance id and check his data in the Walrus.After executing these steps successfully the cloud controller will direct the node controller to make an instance for a new client or open an instance for the existing client.

**6.3.2 Subsystem and Interface Design**

Considering each subsystem separately

**Cloud Controller**

Cloud controller is the core of Cloud Computing and the most important requirement is that it should be up all the time.



Fig.11 High Processing Servers

Node Controller

It is the home of instances and it is also the master of all instances as it guides them with all the required functionalities.



Fig.12 Virtually Enabled Servers

**Software Interface**



Fig.13 Admin Interface

**6.3.3 DESIGN DIAGRAMS**

* **Use Case Diagram**

****

Fig. 14 Use Case Diagram

The Use Case represents the overall scenario of the system. A scenario is nothing but a sequence of steps describing an interaction between the user and a system. CMP’s use case consists of three actors namely, Client, Cloud Controller and Node Controller. The actions performed by the various actors are shown in the figure.

* **Class Diagram:**

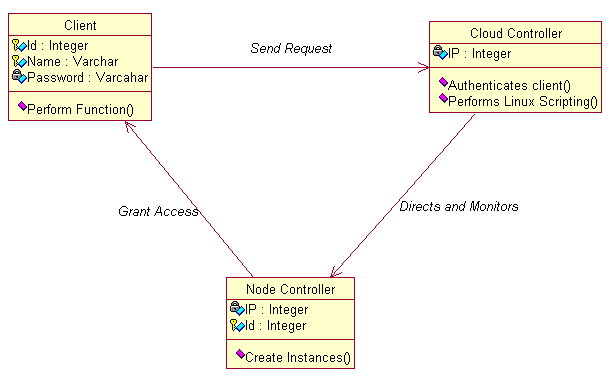


Fig. 15 Class Diagram

A class is a collection of objects that share a common structure and common behavior. The data components of the class are called data members and the function components of the class are called member functions of methods. CMP’s class diagram is as shown above.

* **Activity Diagram**

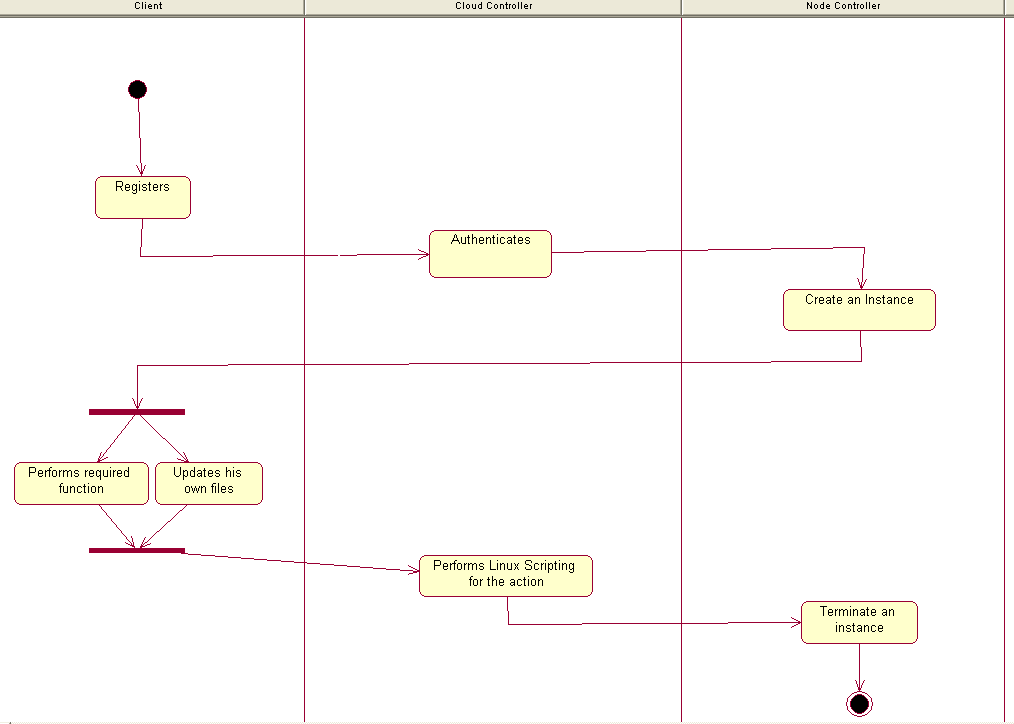
****

Fig. 16 Activity Diagram

Activity diagram is a graphical representation for representing the flow of interactions between specific scenarios. It consists of forks and branches. It also consists of various swimlanes as shown above. The various activities in our project are explained above.

* **Sequence Diagram**

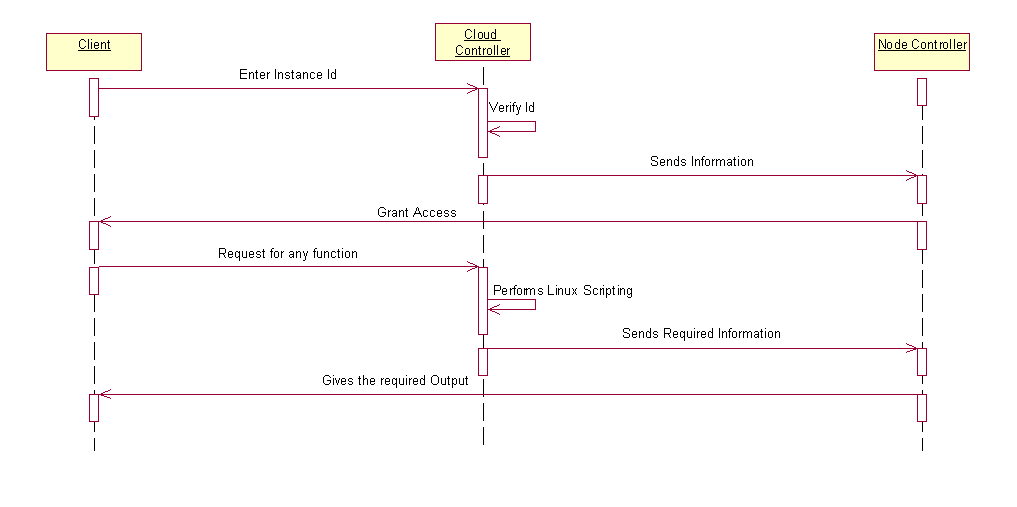
****

Fig. 17 Sequence Diagram

Sequence diagram is a time oriented view of the interaction between objects to accomplish a behavioral goal of the system. It shows how the object interacts with other objects. Step by step interactions are shown above.

* **Collaboration Diagram**

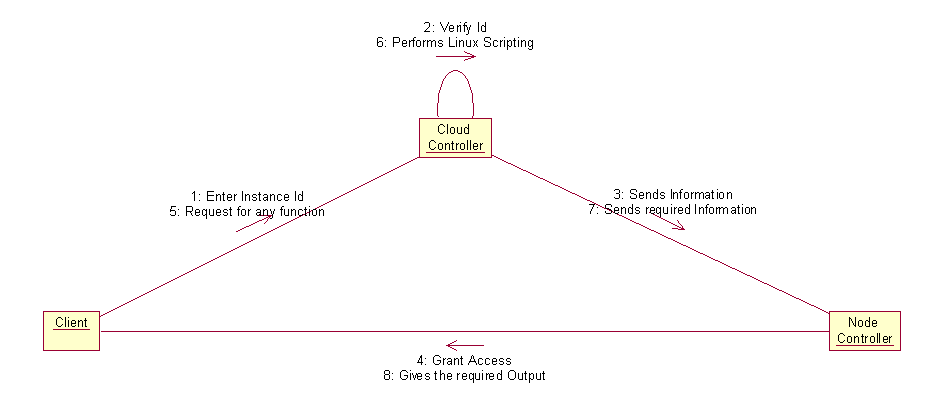
****

Fig. 18 Collaboration Diagram

Collaboration is formed when the objects interact with each other to perform some task. The collaborations between various objects is clearly explained with the help of a collaboration diagram which is as shown above.

* **Statechart Diagram**

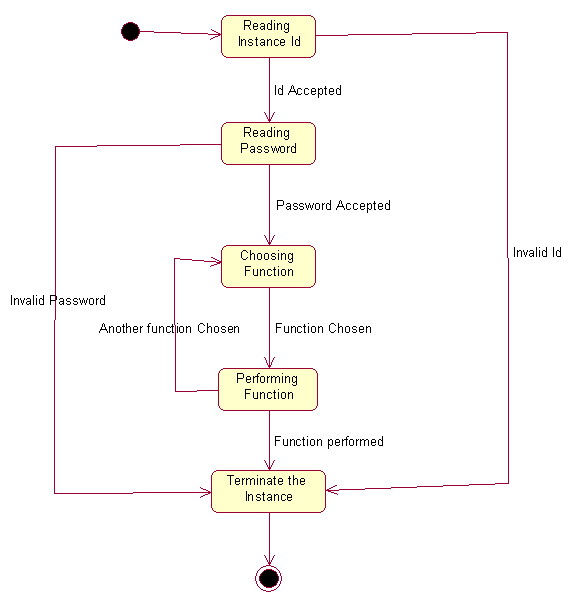
****

Fig. 19 StateChart Diagram

State diagrams are used to graphically represent the states of the system. During the processing the objects change their state in response to events and this can be modeled as the state chart diagram. It is also called as dynamic models. The various state possible in our system are as above.

* **Component Diagram**

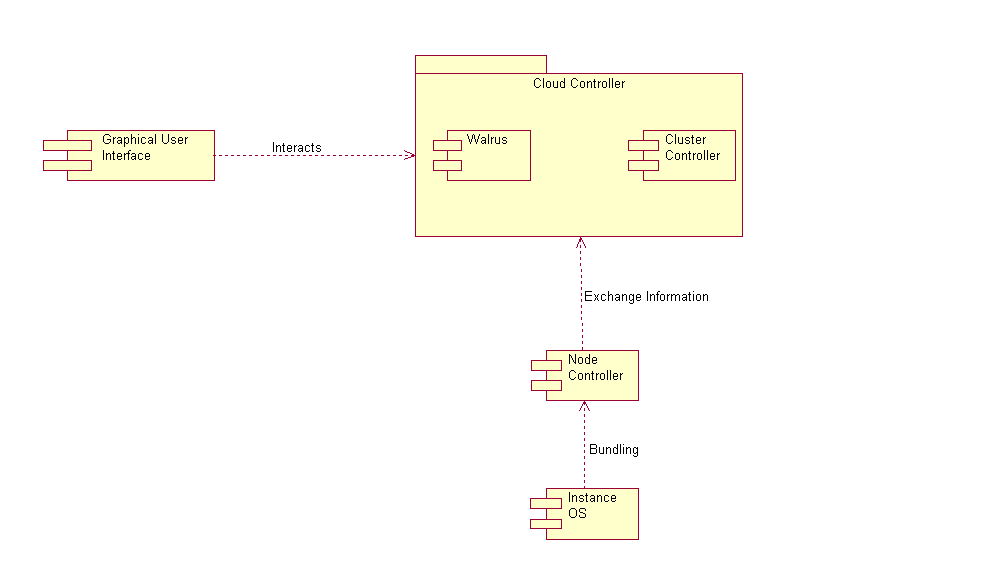
****

Fig. 20 Component Diagram

A component diagram depicts how components are wired together to form larger components and/or software systems. They are used to illustrate the structure of arbitrarily complex systems. The various components are as shown above.

* **Deployment Diagram**

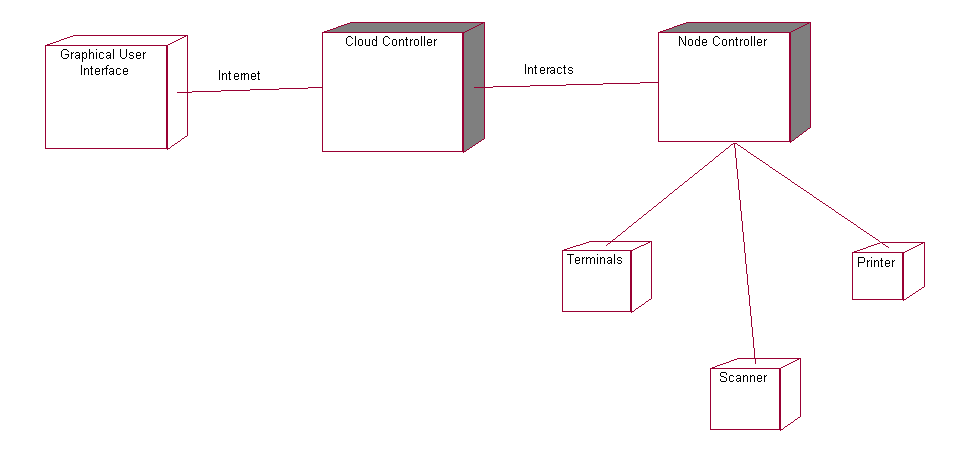
****

Fig. 21 Deployment Diagram

A deployment diagram in the unified modeling language models a physical deployment of artifacts or nodes. The GUI will connect to Cloud Controller via the Internet which in turn interacts with the Node Controller. The terminals, scanners, printers can be connected to Node Controller.

**6.4 OTHER NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions.

The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

**6.4.1 Performance Requirements**

Performance is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used. Depending on the context, good computer performance may involve one or more of the following:

Response time:Response time is the time a generic system or functional unit takes to react to a given input. Response time in our system shall be a low at the Client’s end due to less computational time. Response time depends on the server utilization for that particular client.

Throughput: Throughput is the average rate of successful message delivery over a communication channel. This data may be delivered over a physical or logical link, or pass through a certain network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot. In our project, packets are transferred via the internet between our GUI and the Cloud Controller.

Utilization of computing resource:A resource, or system resource, is any physical or virtual component of limited availability within a computer system. Every device connected to a computer system is a resource. Every internal system component is a resource. Virtual system resources include files, network connections and memory areas. Optimum utilization of computer resources shall be achieved in our project.

Availability: Availability refers to the ability of the user community to access the system, whether to submit new work, update or alter existing work, or collect the results of previous work.A user in our case is eligible to access their accounts and carry out the desired transactions of requesting for any required action round the clock (i.e. 24\*7) which leads to high availability.

**6.4.2 Security Requirements**

Security is the degree of protection against danger, damage, loss, and crime. Security, as a form of protection, is structures and processes that provide or improve security as a condition. Security can be defined as "a form of protection where a separation is created between the assets and the threat"

*Internal Security:*

Internal security deals with the security provided to the data that needs to be transmitted by restricting its access.

Data in our case shall be provided security by providing a private key..

*External Security:*

External security deals with the security provided to the communication path between the cloud controller and the GUI through which packets are transferred.

As our project is based on Cloud Management Platform, so the security is provided by the private cloud architecture itself through mechanisms like Authentication mechanisms such as the Salt algorithm.

**6.4.3 Software Quality Attributes**

The attributes desirable from our software are:

*Availability:*

Is it available when and where I need to use it?

User can access his instance at any time convenient to him/her and request for any action, i.e. 24\*7 provided he is authorized user.

*Efficiency:*

How few system resources does it consume?

Since we have incorporated Performance Monitoring tab, it acts a resource allocator and thus makes optimum use of the available resources.

*Flexibility:*

How easy is it to add new capabilities?

If there is any upgrade/update to our application, client performance would not be hampered and thus it is very flexible.

*Integrity:*

Does it protect against unauthorized access, data loss?

There are various techniques/mechanisms/algorithms used for preventing unauthorized access such as Instance Identification, Private Key, Salt, etc.

*Maintainability:*

How easy it is to correct defects or make changes?

If there is any defect or error in the whole process, it is the admin’s responsibility to foresee it in advance and take the required action to avoid it by following some predefined technique.

*Portability:*

Can it be made to work on other platforms?

Our GUI will support any private cloud as it is independent of the platform.

*Reliability:*

How long does it run before experiencing a failure?

It keeps on working unless and until the core i.e the cloud controller goes down due to some technical glitch.

*Reusability:*

How easily can we use components in other systems?

Components of the application can be reused in other system to serve the purpose of saving time, efforts and money.

*Robustness:*

How well does it respond to unanticipated conditions?

Any unanticipated condition or action will be rejected by the cloud controller or the admin.

*Usability:*

How easy is it for people to learn or to use?

Providing an interactive GUI based application will serve the users with ease of learning and ease of using the cloud.

**6.5 SPECIAL USER REQUIREMENTS**

**6.5.1 Backup and recovery**

There might be a case when the cloud controller goes down. Who will serve the requests?

Eucalyptus High Availability (HA) architecture provides us with a secondary Cloud Controller which starts working when the primary Cloud Controller goes down due to some technical fault. And even this shifting of control happens dynamically giving the client the required data without even delaying for a minute. The possibility of both the Cloud Controller being down is very less and hence HA architecture is apt for private cloud.

**6.5.2 User Training**

The user training is not required much in CPM. Any person who knows how to operate a normal browser with regards to clicking on the required tab will be able to use it perfectly. The entire functionality will be guided by our GUI which will be self explanatory.

**7.0 IMPLEMENTATION**

**7.1 GRAPHICAL USER INTERFACE**

**7.1.1 LOGIN PAGE**

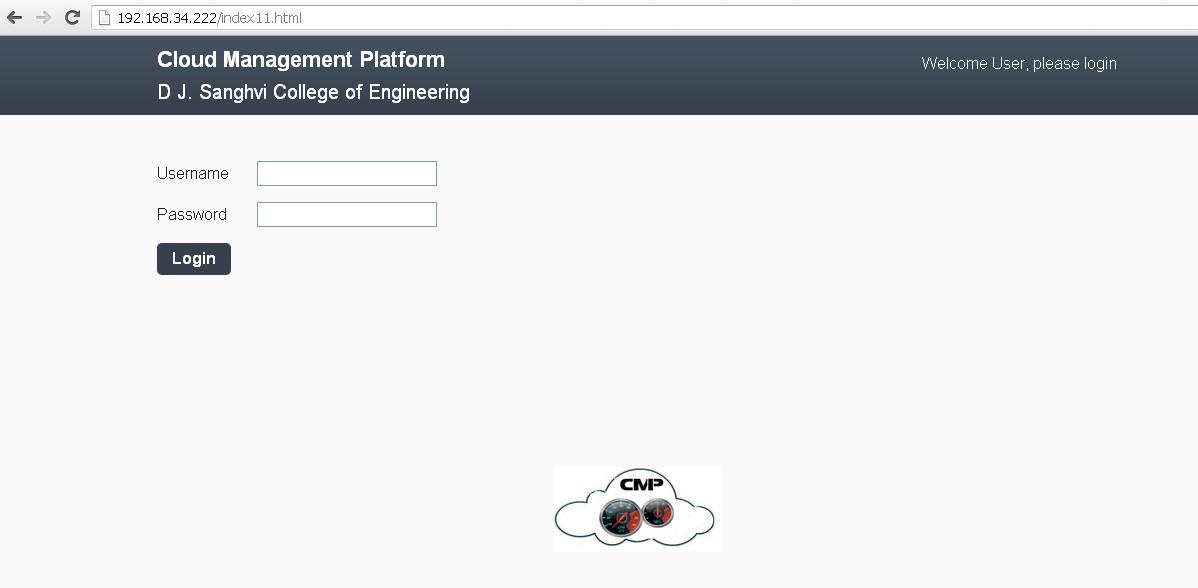
****

Fig. 22 Login Page

**7.1.2 LAUNCH AN IMAGE**

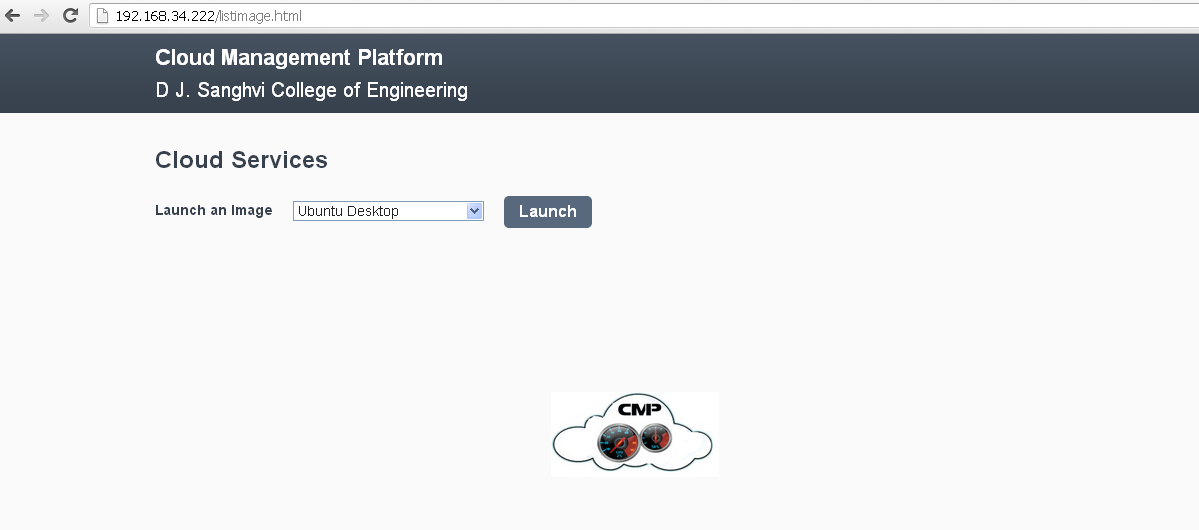
****

Fig. 23 Launch an Image

**7.1.3 RUNNING INSTANCE**

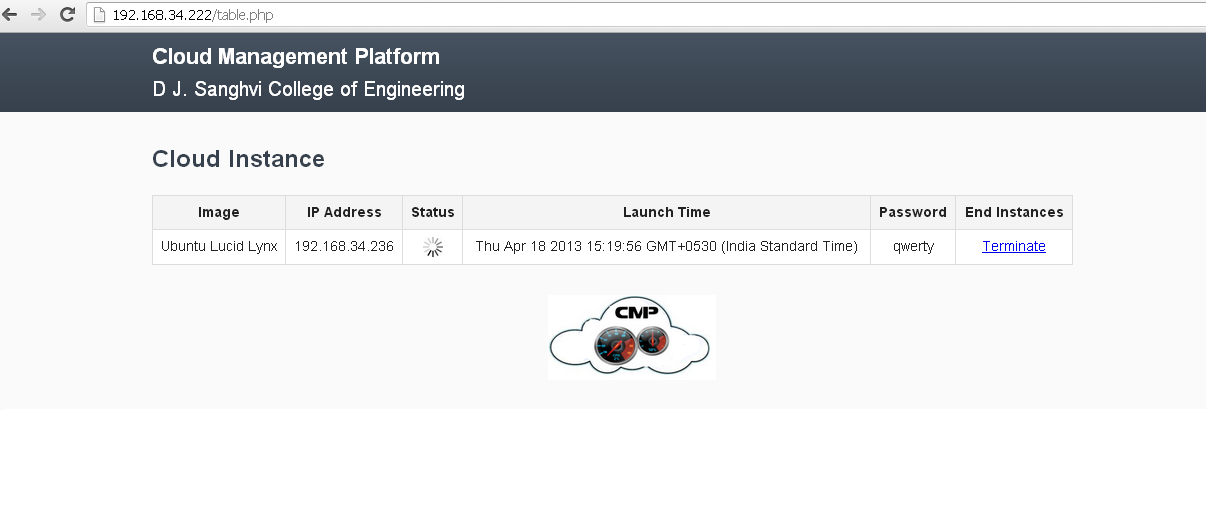
****

Fig. 24 Running Instance

**7.2 CODE**

**7.2.1 PHP CODE TO EXECUTE BOTO FILES**

<?php

$result = system("python botorun.py");

echo $result;

?>

**7.2.2 CONNECTION OF CLOUD WITH BOTO**

import boto

import boto.ec2

region = boto.ec2.regioninfo.RegionInfo(name="eucalyptus",endpoint="192.168.34.222")

connection = boto.connect\_ec2(aws\_access\_key\_id="WKy3rMzOWPouVOxK1p3Ar1C2uRBwa2FBXnCw",aws\_secret\_access\_key="2WdSteXGiF5P3UQou0w4weyLIlNxWzeYbLjw",is\_secure=False,region=region,port=8773,path="/services/Eucalyptus")

print connection

**7.2.3 RUNNING AN INSTANCE USING BOTO**

connection.run\_instances('emi-E1A7108A', min\_count=1, max\_count=1,key\_name='mykey',instance\_type='c1.medium')

**7.2.4 TERMINATING AN INSTANCE USING BOTO**

instances = connection.get\_all\_instances()

r = instances[0]

for inst in r.instances:

a = inst.id

connection.terminate\_instances(a)

**7.2.5 PARAMETERS OF INSTANCES**

from pprint import pprint

from boto import ec2

reservations = connection.get\_all\_instances()

print reservations

instances = [i for r in reservations for i in r.instances]

for i in instances:

pprint(i.\_\_dict\_\_)

**7.2.6 MONITORING AN INSTANCE AND ITS METRICS**

import boto.ec2.cloudwatch as cw

connection.monitor\_instance('i-3C7307B2')

cw\_conn = cw.connect\_to\_region('eucalyptus')

metric = cw.list\_metrics(dimensions={'InstanceId': 'i-52A90912'}, metric\_name='CPUUtilization')[0]

**7.2.7 LINUX COMMANDS**

**7.2.7.1 htop**

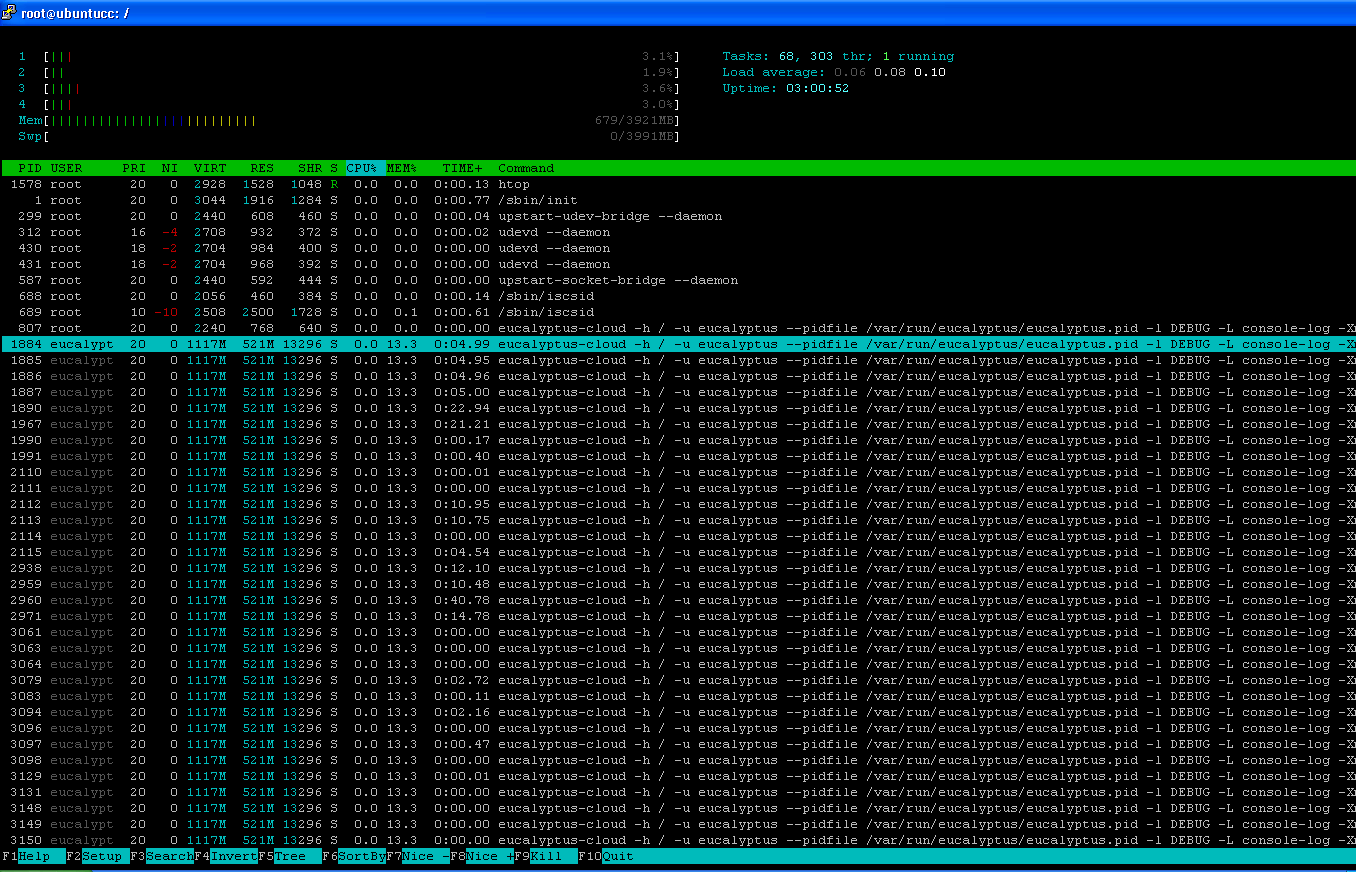
****

Fig.25 htop

**7.2.7.2 iptraf**

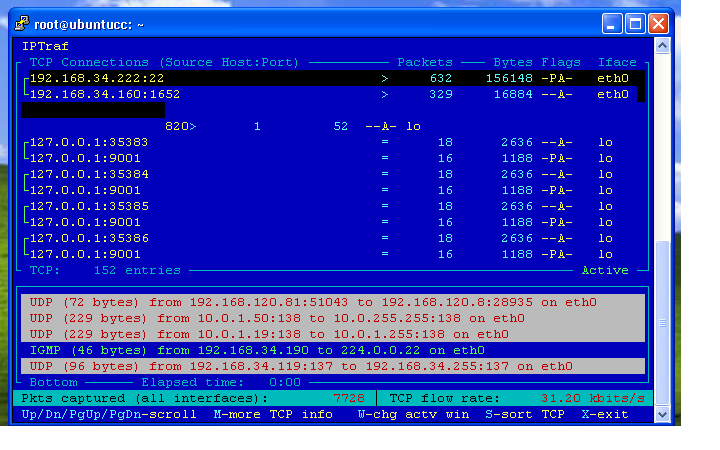
****

Fig. 26 iptraf

**8.0 Project Testing:**

Testing is the process of validating and verifying that an application:

* meets the requirements that guided its design and development,
* works as expected,
* can be implemented with the same characteristics,
* Satisfies the needs of stakeholders.

**8.1 Test plan**

A test plan is a systematic approach to testing a system. The test plan approach that has been used in our project includes the following:

1. *Design verification or Compliance test:*

This testing was performed during the development and approval stage of the product on each of the module.

2. *Acceptance or Commissioning test:*

This testing was performed at the time of delivery and installation of the software.

**8.2 Test Coverage**

The design verification tests were performed at the point of reaching every milestone. Test areas included testing of various modules such as highlight, add question, update keywords, statistics calculation, graph generation etc.

**8.3 Test Methods**

For each module, corresponding outputs were checked in the view of .NET Output Console. For testing highlight module, the output produced from running the VB.NET code was checked with the test data set. The question and keywords modification module was verified with the database. Statistics calculation module was tested using online Word Counter.

**8.4 Test Responsibility**

The team members working on their respective modules performed the testing of those modules. Test responsibilities also included, the collection of data, and decision on how that data should be stored, used and reported.

**8.5 Test cases**

**8.5.1 LOGIN MODULE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID. | Step Description | Expected Result | Actual Result | Pass/Fail |
| 1 | User does not enter username | Prompt user to enter username | Message box to enter username is displayed | Pass |
| 2 | User does not enter password | Prompt user to enter password | Message box to enter password is displayed | Pass |
| 3 | User enters wrong username/password | Prompt user to enter correct credentials | Unsuccessful login message is displayed | Pass |
| 4 | User enters correct username and password | Login is successful | Successful login message is displayed | Pass |
|  |  |  |  |  |

Table 5: Login Module

**8.5.2 INSTANCE MODULE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID. | Step Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Nodes not detected | Display Resources | No resources | Fail |
| 2 | Launch Instance | Run-Instance | Not Enough Resources | Fail |
| 3 | Starting an Instance | Assign IP address | Not enough IP addresses | Fail |
| 4 | Running an Instance | Status: Running | Successful instance running | Pass |

Table 6: Instance Module

**8.5.3 Network Module**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID. | Step Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Cloud Controller Down | Accessing Cloud is possible | No Cloud Controller found | Fail |
| 2 | Node Controller down | To Run Instances | No Node Controller found | Fail |
| 3 | Node Registration | Available Resources | Node Controller not reachable | Fail |
| 4 | ICMP exchange+ Resources | Messages + Resources | Network : Up | Pass |

Table 7: Network Module

1. **MAINTENANCE**

**9.1 User /Technical Manual**

Admin

* Enter the correct secret key and access key for cloud connection.
* Update the client details and maintain the requested instances.

User

* Enroll by entering the SAP id and name on our login page.
* One you are verified as a genuine user, the client can enjoy and use all the available cloud services just by using our Cloud Management Platform.
  1. **Constraints for use of Cloud**
* 24\*7 LAN connection required for the constant connectivity between Cloud Controller, Node Controller and the Client.
* If at some time when the client requests for an instance and if there are not enough resources, he/she will have to wait until the resources are made available.

**10.0 Conclusion and Future Scope**

As stated above, private cloud is a next big thing in IT World because of all the advantages and the flexibility it offers. Thus it generates the necessity for having a cloud management platform that can efficiently manage the cloud and even enhance the user experience since it shields the user from the complicated private cloud architecture and also from the performance monitoring . The above mentioned things have been incorporated in our GUI, thus making it an ideal cloud management platform. Regarding performance monitoring which is a great helping option to the client, we will make a database of all the parameters and will check each of their values in each and every possible scenario to explore each and every possibility after installing the cloud in our private network. For extreme efficiency and accuracy, we would give these values of database to the associative rule data mining technique which would assess each of the cases and would give the most optimum result or solution in case of some problem. In case any scenario exceeds the threshold decided by the data mining technique it will suggest or throw a warning to the admin to take the required actions to solve it.

**11.0 APPENDIX**

Boto - It is a python interface for the eucalyptus running instances.

Bundle – Making kernel, machine and ramdisk image of an operating system.

Credentials – It consists of the secret key and access key automatically generated by the Eucalyptus Cloud unique for each cloud.

Emi – It is the machine image of the operating system.

**12.0 REFERENCES**

[1] Hamzeh Khazaei, Jelena Misic and Vojislav B. Misic,. Performance Analysis of Cloud Computing Centers Using M=G=m=m þ r Queuing Systems*. Parallel and Distributed Systems .IEEE Transactions* vol.23, issue: 5, pp. 936-943, 2012.

[2] Wendy Ellens, Miroslav Zivkovi´c, Jacob Akkerboom, Remco Litjens, Hans van den Berg. Performance of Cloud Computing Centers with Multiple Priority Classes. C[loud Computing (CLOUD), 2012 IEEE 5th International Conference](http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6253102) , pp. 245-252

[3] Hamzeh Khazaei, Jelena Misic and Vojislav B. Misic. Performance Analysis of Cloud Centers under Burst Arrivals and Total Rejection Policy. IEEE Globecom 2011.

[4]http://www.infosys.com/engineering-services/features-opinions/Documents/cloud-performance-monitoring.pdf

[5] http://www.eucalyptus.com/docs/3.1/ig-3.1.0.pdf.

[6]http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[7]http://www.rightscale.com

[8]buildyourownprivatecloudusingubuntu10-04eucalyptusenterprisecloudcomputingplatform-by-animesh-das.pdf

[9] Guidelines-to-BuildingPrivateCloud-Infrastructure-Technical-Report.pdf

[10] Guidelines-to-Private-Cloud-Companion-BookLet-1.pdf

[11] Guidelines-to-Private-Cloud-Companion-BookLet-2.pdf

[12] http://help.ubuntu.com/community/UEC/CDInstall

[13] <http://boto.readthedocs.org/en/latest/ref/ec2>

[14]http://cssoss.wordpress.com/2010/05/10/eucalyptus-beginner’s-guide--uec-edition-chapter-4-image-management