A Resource Provisioning Technique for Memory-intensive Cloud Applications

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Asmita Pandey
(Roll No. 801032001)

Under the supervision of:
Dr. Inderveer Chana
Associate Professor (CSED)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, — **"A Resource Provisioning Technique for Memory-intensive Cloud Applications"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Inderveer Chana* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature

Asmita Pandey

(Asmita Pandey)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Inderveer 02/07/12

(Dr. Inderveer Chana)

Computer Science and Engineering Department,
Thapar University,
Patiala

Countersigned by

Amit K. Verma 02/July/2012.

(Dr. Maninder Singh)
Head
Computer Science and Engineering
Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgment

It is a great pleasure for me to acknowledge the guidance, assistance and help I have received from Dr. Inderveer Chana. I am thankful for her continual support, encouragement, and invaluable suggestions. She not only provided me help whenever needed, but also the resources required to complete this thesis report on time.

I am also thankful to Dr. Maninder Singh head CSED for his kind help and cooperation.

I would also like to thank all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of my thesis work.

I would like to say thanks for support of my classmates. I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

I am highly grateful to my parents and brother for the inspiration and ever encouraging moral support, which enabled me to pursue my studies.

Asmita Pandey

Asmita Pandey

801032001

ABSTRACT

Cloud Computing is gaining popularity in IT industry due to the increase in demand of the computing resources as a service. Cloud Computing reduces the operational and computational cost and thus large companies get benefitted in a manner as they need to pay for as much amount of resource as they need to carry out their present business. With the increase in the requirement they can pay more and get more resources for them, and thus according to their need they can scale up and scale down the resources. Cloud service users require the specific Quality of Service (QoS) in order to meet their objective and sustain their operations on the basis of the Service Level Agreements (SLAs).

Resource provisioning in cloud computing deals with how the resources may be allocated to the application mix such that the SLAs of all applications are met. SLAs may vary from user to user and in order to fulfill the specific SLA it is necessary for the cloud provider to manage the resources efficiently. Resource provisioning allows the users and providers to access the specified resources according to availability of the resources in Cloud as per their request.

There is a need to improve the optimization and allocation of cloud resources. Efficient algorithms for resource optimization are required to be developed for this purpose. There is also a need optimize the resources for memory intensive Cloud applications. In this thesis the existing techniques of resource provisioning on Cloud have been compared and a technique to optimize the memory intensive applications on Cloud infrastructure has been proposed. The validation of the proposed techniques is done on the Eucalyptus Testbed which helps in the scaling of the resources. The proposed technique has been validated through RUBiS application deployed on EUCALYPTUS cloud set up. The experimental results demonstrate that the proposed technique helps to scale up and scale down the cloud infrastructure as per user requirement and thus providing efficient resource provisioning.

Keywords: *Resource Provisioning, Cloud Computing, Optimal Resource Usage.*

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1
Introduction

Cloud computing has eventually emerged from Grid computing. Computing pioneer John McCarthy predicted in 1960's that "computation may someday be organized as a public utility". In the mid-1990s, the term Grid was coined to describe technologies that would allow consumers to obtain computing power on demand. The Cloud Computing is similar to Grid Computing in a manner that it also aims to reduce the cost of computing, increase reliability, and increase flexibility by transforming computers from something that we buy and operate ourselves to something that is operated by a third party. However, now the scenario is that these days with the increase in analysis of massive data, there is an increased demand for computing. In Shanghai on 15th Nov 2007 there were plans for "Blue Cloud," a series of cloud computing offerings that will allow corporate data centers to operate more like the Internet by enabling computing across a distributed, globally accessible fabric of resources, rather than on local machines or remote server farms. Blue Cloud was built on IBM's expertise in leading massive-scale computing initiatives. It was based on open standards and open source software supported by IBM software, systems technology and services. The datacenter hardware and software is called a Cloud [1]. Cloud Computing can be defined as: *A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet* [2].

1.1 Classification of Cloud

Cloud Computing can be classified into following types:

i) On the basis of Resources provided by the cloud;

ii) On the basis of their ownership.

The cloud service model is categorized into three types on the basis of resources provisioned as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [3,4]. *Infrastructure as a Service (IaaS)* [5] provisions are hardware, software, and equipment (mostly at the unified resource layer, but can also include part of the fabric layer) to deliver

software application environments with a resource usage-based pricing model. Infrastructure can scale up and down dynamically based on application resource needs. Typical examples are Amazon EC2 (Elastic Cloud Computing) Service [4] and S3 (Simple Storage Service) [5] where computing and storage infrastructures are open to public access with a utility pricing model. Eucalyptus [6] is an open source Cloud implementation that provides a compatible interface to Amazon's EC2, and allows people to set up a Cloud infrastructure at premise and experiment prior to buying commercial services [7]. *Platform as a Service (PaaS)* offers a high-level integrated environment to build, test and deploy custom applications [8]. Generally, developers will need to accept some restrictions on the type of software they can write in exchange for built-in application scalability. An example is Google's App Engine [9], which enables users to build Web applications on the same scalable systems that power Google applications. *Software as a Service (SaaS)* delivers special-purpose software that is remotely accessible by consumers through the Internet with a usage-based pricing model [8]. Salesforce is an industry leader in providing online CRM (Customer Relationship Management) Services. Live Mesh from Microsoft allows files and folders to be shared and synchronized across multiple devices.

Based on resource ownership, Clouds can be categorized as Private, Public or Hybrid Clouds. In *Private Clouds*, resources are under the legal and contractual umbrella of the organization and thus permanent applications requiring high control over data, security and QoS are most suitable for private Clouds. *Public Clouds* being owned and managed by third parties; pose high concerns for data security making them suitable for temporary applications. However, *Hybrid Clouds* combine the benefits of both approaches that are appropriate for surge Computing; augmenting private clouds with the resources of the public clouds to accommodate high workload spikes.

Consumers require the specific Quality of Service (QoS) in order to meet their objective and sustain their operations from Cloud Service Provider.

It's a duty of the Cloud Service Provider to consider and meet the different QoS parameter of individual Cloud Consumers and negotiate to the specific Service Level Agreement (SLA). SLA defines the requirements regarding the performance, security, availability and the hike from user perspective. Consumers require the specific QoS parameters in order to meet their objectives and sustain their operations. In order to provide the SLA based service to the client the prediction of resources needed to execute the job offered by the client is necessary.

Virtualization provides improved responsiveness by adding features like resource provisioning, monitoring and maintenance which can be automated and common resources can be cached and reused. All these features of virtualization provide the basis for Clouds to meet the SLA requirements in a business setting, which cannot be easily achieved with a non-virtualized environment in a cost-effective manner as systems would have to be overprovisioned to handle peak load and waste resources in idle periods.

1.2 Cloud Service Provisioning

Due to the advent of the virtualization technology and the wide-spread adoption of Services Computing and Web 2.0 application, computing is becoming a utility. The need to install, maintain and run the application in the customer's own computer has disappeared. These days user wants to use the resources and forget about the deployment problems. Software, platform and infrastructure are being hosted by a provider and made available to the user as a service via Internet. The services are typically deployed on the *service provider* infrastructure and that can be provided as a service to the customer.

Service provisioning over the Internet has been highly extended in computing which eventually gave birth to new concept of distributed computing such as Cluster, Grid and Cloud Computing. Cloud Computing refers to both the applications delivered as services over the Internet, the hardware and systems software in the data centers that provide those services [10]. Cloud computing enables the delivery of software, platforms, computation and storage via a distributed computing paradigm with the help of virtualized application environment that can be easily scaled and rapidly provisioned via Internet. The various distributed technologies as depicted in Figure 1.1 currently exist but the changes in technology, usage and implementation over the past so many years have led to the contextual overlap of these systems.

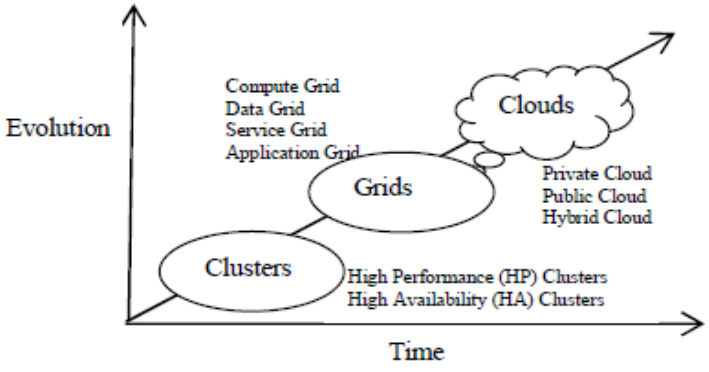

Figure 1.1: Evolution of Distributed Computing paradigms [11]

Now a day's service providers have started providing application as service. This is known as the Service-Oriented Computing (SOC) paradigm [12]. It allows the easy composing of services for building distributed applications. These services are preferred by the service providers that provide implementation, service description and technical support.

From the business point of view, the service provider agrees with its customers for QoS through a SLA. SLA acts as a bridge between service providers and their clients because the user satisfaction levels of the service quality experienced by them have a direct relationship with the profits of service providers [13]. The kinds of services that are commonly requested by a user are much extended, and it can go from word processing software, like Google Docs, to a Web Server that will be available for a week.

It implies a new challenge of managing resources between the applications that have different behaviors and requirements. Managing the resources is one of the biggest problems that has to be solved. Managing a service provider implies a huge effort in terms of each system being used by the customer. During the past few years, dynamic resource allocation for tasks has appeared [14]; nevertheless, these solutions are partial and only take into account one resource level.

In order to maintain the SLA, provider has to reduce the overall cost. Optimization of resources reduces the cost to a greater extent. Resource optimization can be done by predicting the resource requirement and provisioning it accordingly. In order to have better provisioning of the resources, various techniques and methodologies are given by researchers which make resource provisioning in cloud computing a dominant area of research.

Performance, efficiency and availability are the main quality attributes of service. In order to satisfy the promised SLA, these three attributes must be optimal. To make these attributes optimal the service provider must go for provision of all the cloud resources efficiently. There are various resources that are needed to be provisioned. Among these CPU, memory, bandwidth etc. are the major resources to be managed. The allocation of these resources in efficient manner is necessary in order to avoid the overprovisioning and under provisioning of the resources. There is a need of the adequate technique that should manage the resources on the cloud.

The applications that run on the cloud may consist of many tiers depending upon the requirements. A typical Web application consists of three tiers: a Web server tier, an application server tier, and a database server tier. A database tier requires high disk throughput, an application server requires a large amount of CPU time, and a Web server tier requires high

network bandwidth to handle incoming traffic. Therefore a Web application tier depends on one or more of the other tiers. Whenever a tier becomes a bottleneck, the efficiency of dependent tiers also degrades; thus the whole application's performance starts decreasing dramatically.

1.3 Challenges in Cloud Computing

The various challenges that have been identified in the area of Cloud Computing are as follows:

i. Automated Service Provisioning

The objective of a service provider in this case is to allocate and de-allocate resources from the Cloud to satisfy its service level objectives (SLOs), while minimizing its operational cost. These approaches typically involve: (i) Constructing an application performance model that predicts the number of application instances required to handle demand at each particular level, in order to satisfy QoS requirements; (ii) Periodically predicting future demand and determining resource requirements using the performance model; and (iii) Automatically allocating resources using the predicted resource requirements.

ii. Virtual Machine Migration

Virtualization can provide significant benefits in Cloud Computing by enabling virtual machine migration to balance load across the data center. In addition, virtual machine migration enables robust and highly responsive provisioning in data centers. The major benefit of VM migration is to avoid hotspots. Detecting workload hotspots and initiating a migration lacks the agility to respond to sudden workload changes.

iii. Server Consolidation

Server consolidation is an effective approach to maximize resource utilization while minimizing energy consumption in a Cloud Computing environment. Live VM migration technology is often used to consolidate VMs residing on multiple under-utilized servers are set onto a single server, so that the remaining servers can be set to an energy-saving state.

iv. Energy Management

Improving energy efficiency is another major issue in Cloud Computing. The goal is not only to cut down energy cost in data centers, but also to meet government regulations and environmental standards. Designing energy-efficient data centers has recently received considerable attention. A key challenge is to achieve a good trade-off between energy savings and application performance.

v. Data Security

Data security is another important research topic in Cloud Computing. Since service providers typically do not have access to the physical security system of data centers, they must rely on the infrastructure provider to achieve full data security. Even for a virtual private Cloud, the service provider can only specify the security setting remotely, with-out knowing whether it is fully implemented.

vi. Scaling Quickly

Scaling of the resources is very important process for the service provider as well as the service receiver. It is important in a manner to facilitate the desired amount of the resources to fulfill the SLA. This includes the adequate resource provisioning of the resources of the service provider on the cloud.

1.4 Research Motivation

There is a need for the efficient resource provisioning techniques so that the SLA requirements can be met with minimum resource allocation, thus leading to optimum resource utilization. It can be done with the help of scale up and down of the resources on the basis of predictions. There is still a need of optimizing the multiple resources provisioned on the cloud.

A suitable approach for the provisioning of resources will help the cloud providers to manage the resource easily on the cloud. It will also help in the proper utilization of the resources which in turn will increase in the revenue of the cloud providers by reducing the wastage of resource and SLA violation.

1.5 Thesis Organisation

In this master thesis the need of Cloud Computing, its interdependent parameters, available resources and procedure adopted is already described in introduction part of Chapter 1. The rest of the thesis is organised as follows:-

Chapter 2 This chapter describes in detail the literature survey done in the area of resource provisioning in Cloud Computing. The various techniques used so far in this area has been has been concluded in this chapter.

Chapter 3 In this chapter gap analysis is done and the drawback of the existing technique is presented.

Chapter 4 This chapter describes in detail the UML diagrams, proposed model and component diagram of the solution of the problem.

Chapter 5 This chapter focuses on the implementation of the proposed technique and presents the experimental results

Chapter 6 This chapter presents the conclusion and future scope of the work done so far.

Chapter 2

Literature Survey

This chapter presents some key concepts that will be used during the rest of the thesis. The first and the most important concept is virtualization; that allows abstracting the machine management to software management. The second one is Resource Provisioning; need and resource dependency at various service level of cloud. Finally some previous work done so far in the area of resource provisioning which is surveyed to present the various technical advancements that helped in the foundation of resource provisioning in cloud computing.

2.1 Virtualization

Virtualization is based on creating an interface that hides implementation issues through encapsulation. Virtualization is used since 1960s as software abstraction layer that partitions a hardware platform into virtual machines which simulate the underlying physical machine and allows running unmodified software [24]. This mechanism provides a way to multiplex application usage to users sharing processor time.

Software development takes great benefit of virtualization and one of the biggest is the development of cloud computing toolkit to provide various services to the consumer. Having a complete profiled system permits easy management of services and resources. Another improvement that can be obtained with virtualization is migration. An application (or the complete operating system) can be migrated to another machine. This aspect is one of the features of application virtualization, full virtualization, Para-virtualization and library virtualization. With these techniques an application can be moved to a different hardware without any modification. In addition, some of these methods allow live migration; in other words, moving an application to another place while it is being executed. For deploying various services needed by the client, there is a need to deploy the desired machines. Virtualization had made it possible for us to deploy several VMs in a single machine by reducing the hardware needed and saving deploying time [28]. To achieve the hardware and software virtualization various hypervisors have come into existence. Some of them are KVM, Xen, VMware, VirtualBox.

With the help of virtualization technology there are various cloud computing platforms that have been developed these days. These computing platforms made use of the hypervisor technology

which helped in the virtual partitioning of the resources, and thus enabled the management of the resources. The various platforms that have been come across, their advantages and various other features, which guided us for selecting the Cloud Platform to precede the work is described below.

2.2 Cloud Computing Platforms

Cloud platforms are generally categorized into two; i) open source cloud platforms; ii) closed source platforms. Open source is a term for any program where the source code is available for editing, use or modification by users and developers to fit their own demands and requirements.

Table 2.1: Comparison of Cloud Computing Open Source Virtual Resource Sets

Cloud Tool	**EC 2**	**EBS**	**S3**	**Drawback**	**Virtualization**	**Advantage**	**Service**
Eucalyptus [15]	yes	Yes	yes	Does not allow to move running instances between the connected nodes	KVM, Xen, VMware	Walrus storage service is compatible with Amazon EC2. It consists of Cloud Controller, Cluster Controller and Node Controller.	IaaS
OpenNebula [17]	Partially	No	No	Does not include a storage service.	KVM, Xen, VMware, VirtualBox	It allows to migrating the running instances between the connected nodes. Provides High Performance Computing as a Service.	IaaS
Nimbus [18]	Partially	No	Partially	does not feature an EBS-compatible storage service	Xen and KVM	Cumulus Storage	IaaS
CloudStack [19]	Partially	No	No	Focuses mainly on data center computing	KVM, Xen, VMware, VirtualBox	Contains Management Server and the Compute Nodes.	IaaS
OpenStack [20]	Yes	No	No		UML	It is made up of three components namely Nova which is an infrastructure as a service suite, swift which is a scalable redundant storage system and open image service which is used for registering, discovering and retrieving virtual machine images.	IaaS

Closed source is a term for any program where only the final product is available to the public or user. Their source code is not available for modification by users.

The platforms that are going to be discussed here will be categorized into open source and closed source. Table 2.1 summarizes the various cloud platforms.

From the above table it can be concluded that Eucalyptus is the platform that can easily be deployed for the testing of our applications. It can be easily scaled according to the application requirement.

2.3 Need of Resource Provisioning in Cloud

Allocation of the adequate amount of resources to the application while considering the SLA associated with the application is still one of the major challenges in Cloud Computing. This can be achieved by dynamic provisioning of the resources. Dynamic provisioning enables additional resources such as server to be allocated to an application on-the-fly to handle the increased workload. When the workload increases, the policing enables the application to temporarily turn away the excess requests while additional resources are being provisioned [21]. Various reasons that motivated to develop different techniques for resource provisioning are:

- The increasing difference between these resource allocations and application requirements which magnifies resource underutilization thereby leading to increase in cost [22].

- To deal with the problem of under provisioning of resources thus missing potential customers and revenue [23].

- With the advent of the virtualization technology which is a software method and that allows the use of the same physical resources by different applications, it is easy to optimize the resources. This results in a significant reduction of power consumption on server, which facilitated scaling an easy operation [24-28].

- Carefully provisioned resources are easy to manage thereby reducing the efforts of the provider.

2.4 Possible Resources to be Provisioned for Cloud

The cloud provides services to the users at different level of abstraction which has been classified by NIST as service models and when they are deployed on cloud is referred as cloud deployment model [29]. The resources used by each of the cloud provider as a service is summarized in the table 2.2.

Table 2.2: Resources to be Provided on Various Service Models

Service models	Resources to be Provisioned	
	On cloud by provider	**By cloud to client**
Software as a Service (SaaS)	CPU and memory for n number of tiers.	Cloud application
Platform as a Service (PaaS)	Application engine, platform physical and resources like CPU and memory.	Cloud software
Infrastructure as a Service (IaaS)	Computational, storage and communication resources for operational infrastructure.	Cloud software infrastructure

Software as a Service (SaaS) makes use of the web browser to provision the software to the client through the web browser which has been developed by the other user. The resources that are needed to be provisioned are mainly CPU and memory by the provider. Some of the example of SaaS are Gmail [30], Google Docs [31], Amazon [32], The Weather Channel [33] etc.

Platform as a Service (PaaS) provides application engine for running a programming language and tool in order to facilitate the provider a platform for the deployment of various applications. Google App Engine [34] and Microsoft Azure [35], Facebook [36] are popular examples that use the PaaS model for Cloud Computing.

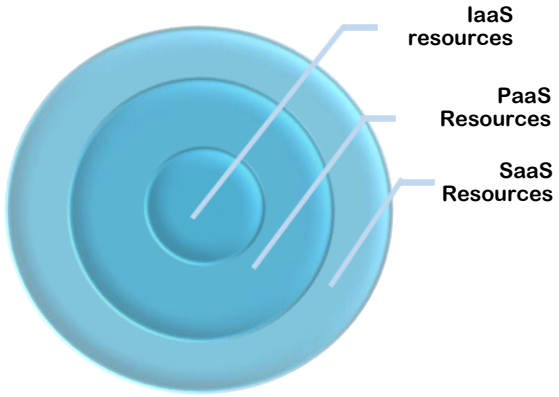

Figure 2.1: Resource Dependency of Cloud Services

Infrastructure as a Service (IaaS) provides computing resources to deploy and run the applications to the customer. It gives much more flexibility than IaaS, and PaaS and allows user to deploy at operating system directly. Amazon Web Services EC2 and S3 are the popular example [37]. Various cloud services have different kind of resource dependency that is common between them. The elasticity of the resources is directly proportional to the type of service offered as shown in Figure 2.1. It is clearly observed that IaaS resources are easier to scale which

allows any kind of software configurations to be deployed on top of them. On the other hand the scaling operation of SaaS is dependent upon PaaS and IaaS, less flexible as compared to PaaS which is only dependent on IaaS as shown in core of Figure 1.2.

2.5 Existing Techniques for Cloud Resource Provisioning

As described above that the provisioning of resources for cloud is a critical barrier and that has to be managed through different service models. Each model has its own limitations. However, attempts have been made to utilize the resources optimally by considering different parameters. In this section, the review of the work that describes the techniques and methodologies used in resource provisioning is presented. The technologies in resource provisioning include the software and hardware management with the help of virtual machine and various prediction methodologies used to evaluate management decisions.

Resource allocation with the help of the placement of virtual machines was carried out by Andreolini *et. al* [38]. They presented an algorithm to reallocate the placement of virtual machines in a cloud for better performance and resource utilization by measuring CPU utilization to identify the over utilized and underutilized machines. The algorithm identifies three sets of machines, most over utilized physical machines as senders, most underutilized physical machines as receivers, and most over utilized virtual machines hosted on the senders as guests to migrate. However, the drawback of this algorithm is that they only considered CPU utilization. Memory utilization was still a challenge while identifying the over utilized and underutilized machines. Thus leaving the memory bounded jobs to create a bottleneck.

The multi-tier applications are usually happened to be hosted on the cloud. Jung *et. al* [39] worked on the optimal configuration of a multi-tier application hosted in a virtualized data centre by developing a queuing network model which pre-deployed the application followed by the identification of the values of required parameters for their model. Previously Jung *et. al* [40] developed an online controller to predict the performance and resource utilization of the multitier applications. The controller used a bin-packing-based algorithm to produce target configurations that exhibit the least amount of performance degradation. A select-reject technique iteratively identifies a physical machine that has enough CPU capacity to host the tier to maximize fault tolerance [41]. It does not host another instance of the same tier. While typical datacentre environments have multiple pools of resources, it considered only CPU leaving back servers, disks and input/output channels.

The generation of hotspots in virtualized data centre is very common due to the increasing workload. Sandpiper [42] is a system for recognition and mitigation of hotspot (overloaded) virtual machines in a Xen-based virtualized data centre [27]. It uses the statistics for physical and virtual machine to create profile for all physical and virtual machine and uses that data to recognize the hotspots. The hotspot virtual machine is mitigated to appropriate physical machine using greedy algorithm. Wood *et. al* [41] studied specifically Xen [27] based virtual environment which might not be applicable to other environments. The method solely uses the black-box technique to eliminate simultaneous hotspot. Gray-box information can improve the memory allocation, references about resource requirement and the performance of system.

Bodik *et. al* [42] worked on predicting system performance by statistical machine learning approach to minimize the number of resources required to maintain the performance of an application hosted on a cloud. The next 5 minute's workload prediction is done using linear regression based on the last 15 minutes of traffic. This prediction is used as an input to a performance model that estimates the number of servers required for handling the predicted workload. The system adjusts the performance model using machine learning techniques such as online training and change point detection. However, the issues of multitier Web applications are not addressed.

Urgaonkar *et. al* [43] presented an analytical model using queuing networks to capture the behaviour of tiers which is based on using a network of queues to represent how the tiers in a multi-tier application cooperate to process requests. The model enables to predict the mean response time for the given workload, for which the model requires several parameters such as the visit ratio, service times, and think time. They processed the log files of tiers to calculate these parameters. They took into account on improving one of the major SLA i.e. response time. However, they studied the variability of the response time with respect of CPU and left to analyse the effect of memory parameter on response time. The author believed that effect of input/output intensive jobs on the model is still a challenge for them.

Villela *et. al* [44] argued that the optimal resource allocation to the application server tier in a multi-tier e-commerce Web application hosting environment will increase the profit of service provider. He devised three methods for allocating a fixed number of servers among an arbitrary set of customers with a variety of traffic demands and different SLA configurations. The first method uses only an estimation of average response time to evaluate costs for the provider. The

second method utilizes an estimation of variance of response times to evaluate costs for the provider. The third method utilized a Poisson process description. The request arrival process was taken as a Poisson process by analysing the traces of a real Web application and formalized the resource allocation problem to maximize the profit of service providers by optimal resource allocation to the application server tier. They have evaluated three different approximation methods for optimal resource allocation using simulations.

Dubey *et. al* [45] aimed to model a distributed multi-tier enterprise application by performing dynamic regression and queuing modelling techniques on the results of experiments performed on distributed multi-tier Web application hosting on virtualized data centres. The experiment was performed using the dayTrader [46] benchmark Web application to evaluate their proposed model system on a Xen-based [27] virtualization platform. However, the author was not able to identify the resource responsible for bottleneck. In one of the experiment CPU was taken as bottleneck which was later proved to be invalid.

Reig.G.*et. al* [47] proposed a scheduler and a prediction system for managing the job resource requirement. The scheduler maintains the record of arrived and pending jobs along with the status of resources and takes help of prediction system for decision making regarding whether to discard the job or to run the jobs on the available resources. Prediction system contains the analytical predictor which worked initially to predict the job resource requirement when the Self-Adjusting predictor is not trained and self-adjusting predictor uses machine learning techniques for predicting the resource requirement for completely different jobs and also overcomes the problem of inadequate accuracy of Analytical Predictor. The author has compared both the predictors and concluded that the self-adjusting predictor is more accurate as compared to the analytical predictor [48].

Jungy *et. al* [49] has presented off-line techniques to generate adaptation policies for multi-tier applications hosted on virtualized data centre. The purpose of an adaptation policy is to provide optimal configurations of an application for the given workload. They generated a queuing model with optimization techniques and decision learning to generate optimal system configuration for a multi-tier application. Their model is able to identify the number of replicas for tiers, tiers placement, and CPU allocation for tiers. The model was implemented upon Xen [27] virtualized environment. However, the memory allocation was not taken into account.

Iqubal *et. al* [50] worked on the multi-tier applications hosted on cloud. Taking into account the web application SLA parameters they developed a technique to provide Response time as SLA to the cloud providers and accomplished through dynamic resource allocation in virtual Web farm. The methodology includes detection of the bottleneck tires that is responsible for lowering the response time and allocating additional resources dynamically to those tires according to the demand. To achieve this they have developed two components VLBCoordinator and VLBManager. The function of VLBCoordinator is to instantiate a new virtual machine and obtains an IP address for virtual machine. The VLBManager on the other hand monitors the logs of the load balancer over the interval of 60 seconds and in case virtual machine exceeds the required response time it asks the VLBCoordinator to instantiate a new virtual machine and VLBCoordinator assigns a new IP address to it. The result depicted that the static allocation of resources cannot provide the SLA guaranteeing a specific response time while on the other hand adaptive allocation solved this problem by offering SLA that enforced a specific response time.

Gong *et. al* [51] proposed another online resource demand prediction scheme to predict dynamic application resource requirement. The prediction system used two techniques; i) signature driven resource demand prediction in which resource usage patterns are discovered, patterns repeat themselves because of the repeated requests and iterative computations thereby prediction can be done with the help of historic usage data of same pattern; ii) state driven resource demand prediction is done for non-repeating patterns which is a statistical state driven approach to capture short-term patterns in resource demand, and uses discrete-time Markov chain to predict that demand for near future. Although the author mentioned that the PRESS [51] is capable of managing memory, I/O, and network usage, but they tested the application exclusively for CPU bounded jobs only.

Later Iqbal *et. al* [52, 53] extended the work to Web application consisting of two tiers, a Web server tier and a database tier, hosted on heterogeneous compute cloud. The various scaling techniques mentioned here were horizontal scaling and vertical scaling. He classified the Web content request into static and dynamic content. A proposed model have been developed and tested on read intensive application for the scaling of web server and database server. The heuristics and application profiling helped to identify the bottleneck tier. The classification of data is done as static and dynamic by reading the log for 60 sec. 95th percentile of the average response time calculation helped to scale the database depending on the saturated response time.

If the static content response time is saturated then scaling up of web server is done otherwise for resolving the dynamic content response time first of all the CPU utilization is checked, if it is saturated then scaling of web server is done else scaling up of database server is performed.

Further Iqbal *et. al* [54, 55] have modelled to scale down the resources to prevent the wastage of resource. It has been named as predictive model that predicts the web server instances and database server instances required for observed workload. The model consisted of a polynomial equation of degree two for the calculation number of web server tier and database server tier which was calculated after putting the values of the number of static and dynamic requests received during that time interval and some regression coefficients that are recalculated after updating the sufficient statistics for all the historical data every time whenever a new observation is received. For a certain interval of time static content response time and dynamic content response time is observed. If static content response time and dynamic content response time is under threshold value for last k intervals then Web server tier and database server tier is scaled down respectively. The vision to provide accurate resources to meet the SLA of response time was achieved to a greater extent and predictive model eliminated the problem of overprovisioning of the resources. However, in all of his work they have not considered any technique regarding how to provision the resources for memory intensive jobs.

The above existing techniques have been summarized in Table 2.3

Table 2.3: Existing Techniques for Resource Provisioning of Multi-tier Applications Hosted on Cloud

Technique	Application Environment	Virtualization Environment	Description	Findings	Resources taken into account
Placement of Virtual Machines [38]	Cloud Platform	Linux and MS virtualized servers	The algorithms identify the critical instances and take decisions without recurring to typical thresholds by reallocating the virtual machines in a cloud.	Significant improvements in terms of selectivity and robustness of the proposed algorithm for sender detection and selection of the most critical guests for migration.	CPU
Queuing Network Model [39]	Multi-tier application platform	Xen	Uses Bin-packing-based Optimization technique to produce target configurations that exhibit the least amount of performance degradation.	The proposed approach provides better availability and maximum throughput than classical approaches.	CPU

Identification and Mitigation of Hotspot [41]	Data center environment	Xen	Sandpiper recognizes the overloaded virtual machines and tries to find the requirement to mitigate virtual machine to appropriate physical machine using greedy algorithm.	It is capable of eliminating simultaneous hotspots involving multiple resources.	CPU, memory, bandwidth
Statistical Machine Learning Approach to Predict System Performance [42]	Amazon's EC2 cloud	Xen	Predicted system performance by statistical machine learning approach to minimize the number of resources required to maintain the performance of an application hosted on a cloud	The system adjusts the performance model using machine learning techniques such as online training and change point detection.	Workload balance on nodes
Analytical model using Queuing Networks [43]	Multi-tier application environment	Kernel TCP Virtual Server	An analytical model using queuing networks to capture the behavior of tiers which is based on using a network of queues to represent how the tiers in a multi-tier application cooperate to process requests.	They took into account on improving one of the major SLA i.e. response time, by studying the variability of the response time with respect of CPU.	CPU
Optimal Resource Allocation using Simulations [44]	E-commerce Environment	Simulated Environment	Devised three methods for allocating a fixed number of servers among an arbitrary set of customers with a variety of traffic demands and different SLA configurations.	These methods offer a low complexity solution that can significantly increase profits.	Servers
Approximate System Performance Model [45]	Distributed environment	Xen	To model distributed multi-tier enterprise application by performing dynamic regression and queuing modeling techniques on the results of experiments performed on distributed multi-tier Web application hosting on virtualized data centers.	Generated models can accurately predict the system behavior and accordingly can support a variety of model-based management techniques.	CPU
Scheduler and a Prediction System for Managing the job resource [47]	SaaS Cloud environment	Xen	A scheduler and a prediction system for managing the job resource requirement.	The author has compared both the predictors and concluded that the self-adjusting predictor is more accurate as compared to the analytical predictor.	CPU, memory
Multi-tier applications hosted on virtualized data center [49]	Data center environment	Xen	The purpose of an adaptation policy is to provide optimal configurations of an application for the given workload.	The proposed model is able to identify the number of replicas for tiers, tiers placement, and CPU allocation for tiers.	CPU
Multi-tier Applications Hosted on Cloud [50]	EUCALYPTUS-based compute cloud	KVM	Technique to provide Response time as SLA to the cloud providers and accomplished through dynamic resource allocation in virtual Web farm.	Static allocation of resources cannot provide the SLA guaranteeing a specific response time while on the other hand adaptive allocation solved this problem.	CPU

PRESS [51]	Cloud Computing	Xen	Signature driven and state driven prediction technique	Resource usage prediction allocations to achieve better service provider profitability	CPU, memory, bandwidth
SLA-Driven Dynamic Resource Management [52,53]	EUCALYPTUS cloud	KVM	A proposed model has been developed and tested on read intensive application for the scaling up of web server and database server	Enables the provider to offer SLAs that define specific maximum response time requirement for multi-tier Web applications by detecting the bottleneck.	CPU
SLA-Driven Adaptive Resource Management [54,55]	EUCALYPTUS cloud	KVM	Model to scale down the resources to prevent the wastage of resources is devised along with the scale up approach discussed in [34,35]	Predictive model eliminated the probability for overprovisioning of the resources to a greater extent.	CPU

2.6 Conclusion

From the data summarized in Table 2.3 it can be concluded that there is still a need to improve the continuous optimization and allocation of resources. Efficient algorithms for resource optimization are required to be developed for this purpose. Provision of response time as SLA is still a challenging factor for cloud providers. Less work has been carried out for memory intensive applications.

The next chapter describes the gaps in the existing techniques of the resource provisioning which will help in the formulation of the problem statement.

Chapter 3

Gap Analysis

The previous chapter described various techniques of resource utilization for provisioning the underutilized and over utilized resources in multi-tier applications. Literature survey shows that lots of work has been done for improving the resource management in multi-tier applications, but there is still a need to improve the continuous optimization and allocation of resources. An efficient resource provisioning algorithm should be dynamic so it can get adjusted to the dynamically changing workload of cloud.

Iqbal *et. al* in [54] proposed a reactive model for scale up operation and a predictive model for the scale down operation. In the reactive model web server tier and database server tier resources were taken into account for scale up operation. However they did not consider the memory (RAM) utilization which has a significant impact on the response time. In the predictive model a regression model that predicts for each time interval t, the number of Web server instances and number of database server instances required for the current observed workload. This model fails to predict the memory requirement. Hence there is a need to propose a model for that can predict the amount of memory required at required in cloud infrastructure.

3.1 Problem Statement

Iqbal *et. al* in [54] proposed a Predictive model for scale down operation which can be described as follows:-

$$n_t^{web} = a_0 + a_1 (h_t^s + h_t^d) + a_2(h_t^s + h_t^d)^2 + \epsilon_t^{web} \quad (1)$$

$$n_t^{db} = b_0 + b_1 (h_t^d) + b_2(h_t^d)^2 + \epsilon_t^{web} \quad (2)$$

where h_t^s and h_t^d are the number of static and dynamic content requests received during interval *t*. Since both static and dynamic resource requests hit the Web server tier, n_t^{web} (the number of Web server instances required, Equation 1) depends on both h_t^s and h_t^d. Since the database server only handles database queries, which are normally only invoked by dynamic pages, n_t^{db} (the number of database server instances required, Equation 2) depends only on h_t^d. After updating the model with sufficient number of requests data the regression coefficients a_0, a_1, a_2, b_0, b_1, and b_2 can be calculated and normalized. After that the Equation 1 and Equation 2 will predict the value of n_t^{web} and n_t^{db} for certain value of h_t^s and h_t^d hitting the cloud.

It is observed clearly that there is no provision in the Equation 1 and Equation 2 to predict the memory usage. Hence in this work a Memory Predictive Model based on the existing model has been proposed.

This thesis aims to develop a model for the optimum utilization of the resources in the cloud environment that will develop the existing model by giving the significance to memory as a resource in various tier of multi-tier applications hosted on the cloud.

Chapter 4

Design of the Proposed model

This chapter discusses the system design and implementation details. A methodology is presented to identify the bottlenecks. When bottlenecks are identified, the system dynamically allocates the resources required by the application to resolve the identified bottlenecks and maintain response time requirements. The system furthermore can predict the optimal configuration for the dynamically varying workload and can scales down the configuration whenever possible to minimize resource utilization.

The content request arriving on the cloud can be described as static content request and dynamic content request. The classification of it is done in Table 4.1 below:

Table 4.1: Classification of Request on Web

Type of Content Request	Resources	Tier
1. Static content request	Static resources	Web Server Tier
	Memory	
	Pre-generated files	
2 Dynamic content request	CPU utilization	Includes mixture of one or more Tier
	Server Side Scripts	
	Memory	

The above resources can be dynamically scaled in accordance with the traffic arriving on the cloud on different time instances. The various scaling techniques for web applications are tabulated in the Table 4.2 below.

Table 4.2: Types of Scaling

Scaling Type	Technique of scaling
1. Horizontal Scaling	Load Balancing
2. Vertical Scaling	1. Migration to a new physical machine
	2. Dynamically increase the hardware allocated

4.1 Memory Predictive model

To determine when to initiate scale down operations for memory, a regression model is proposed that predicts, for each time interval *t*, the memory usage of Web server is given by $f(n_t^{web})$ and number of database server instance $f(n_t^{dB})$ required for current observed workload. The model can be expressed as follows:

$$f(n_t^{web}) = a'_0 + a'_1 r_t^{web} + a_2 (r_t^{web})^2 + \epsilon_t^{web} \quad (3)$$

$$f(n_t^{db}) = b'_0 + b'_1 r_t^{db} + b'_2 (r_t^{db})^2 + \epsilon_t^{dB} \quad (4)$$

Where r_t^{web} and r_t^{db} are the number of requests on web server tier and data base server during the time interval *t* that can be obtained on the basis of the usage statistics. The regression coefficients $a'_0, a'_1, a_2, b'_0, b'_1, b'_2$,can be recalculated after updating the sufficient statistics for all of the historical data every time till a new observation is received.

4.2 Reactive Model for scale up of Memory

The proposed model for the scaling up operation has been depicted in Figure 4.1. It will first of all detect the bottleneck resource and will accordingly scale up the respective tier. Heuristics and active profiling of the Memory and CPU of virtual machine-hosted application tiers for identification of bottlenecks can be used. The system will read the Web server proxy logs for *t* seconds and clusters the log entries into dynamic content requests and static content requests. Requests to resources (Web pages) containing server-side scripts (PHP, JSP, ASP, etc.) are considered as dynamic content requests. Requests to the static resources (HTML, JPG, PNG, TXT, etc.) are considered as static content requests. Dynamic resources are generated through utilization of the CPU and memory may depend on other tiers, while static resources are pre-generated at files available in the Web server tier. Each type of request has different characteristics and is monitored separately for purposes of bottleneck detection. The system will have to calculate the 95th percentile of the average response time. When static content response time indicates saturation, the system will check the RAM first. If it finds that the RAM is saturating it will scale the RAM of the web server tier, otherwise it will scale the Web server tier. When the system determines that dynamic content response time indicates saturation, it obtains the CPU utilization across the Web server tier. If the CPU utilization of any instance in the Web server tier has reached a saturation threshold, it will first of all check the RAM usage of the web server tier. If RAM usage is saturating it will scale the RAM of the tier; otherwise the system scales up the Web server tier; otherwise, it scales up the database tier. Each scale up operation adds exactly one server to a specific tier. If the system statistics the response time requirements for k consecutive intervals, it uses the Memory predictive model to identify any over-provisioned resources and if appropriate, scales down the over-provisioned tier(s). The Memory predictive model is already described in Section 4.1.

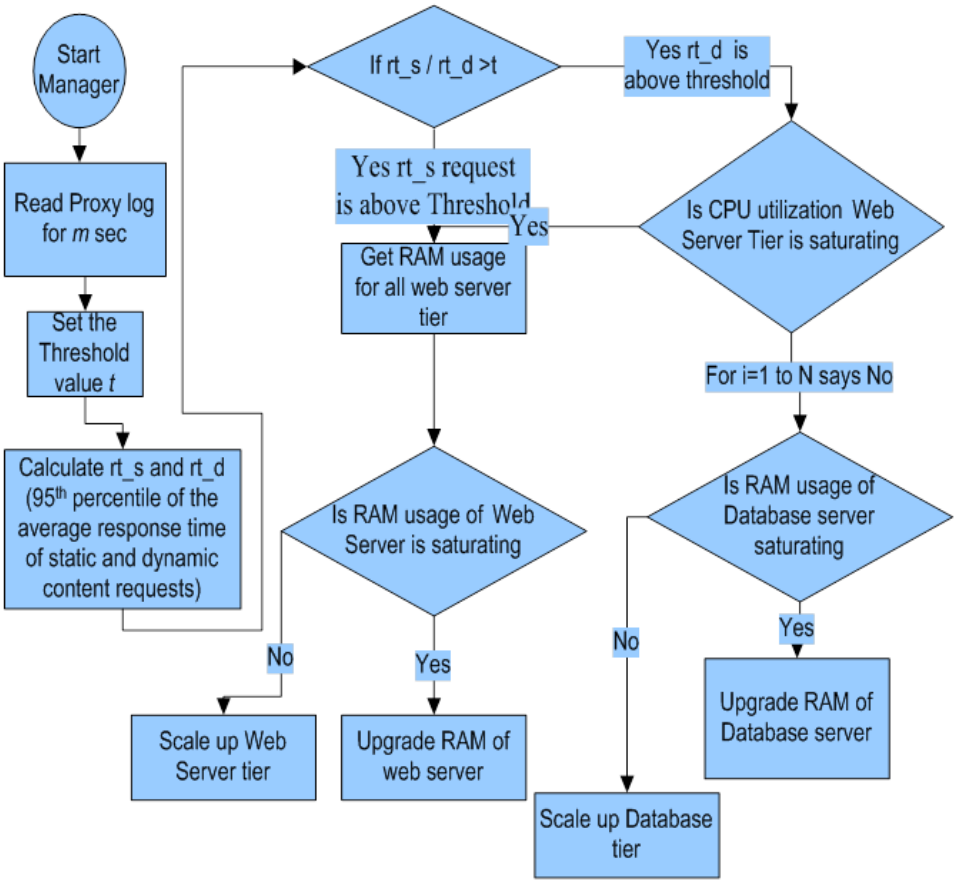

Figure 4.1: Model for Scaling up operation of Cloud Resources

4.3 Proposed Model for scale down operation

The Figure 4.2 depicts the proposed model for the scale down operation. It will first of all detect the bottleneck resource and will accordingly scale it down by making use of the Memory Predictive model discussed in 4.1. The amount of memory that is downgrading depends upon the prediction received form the Memory Predictive Model.

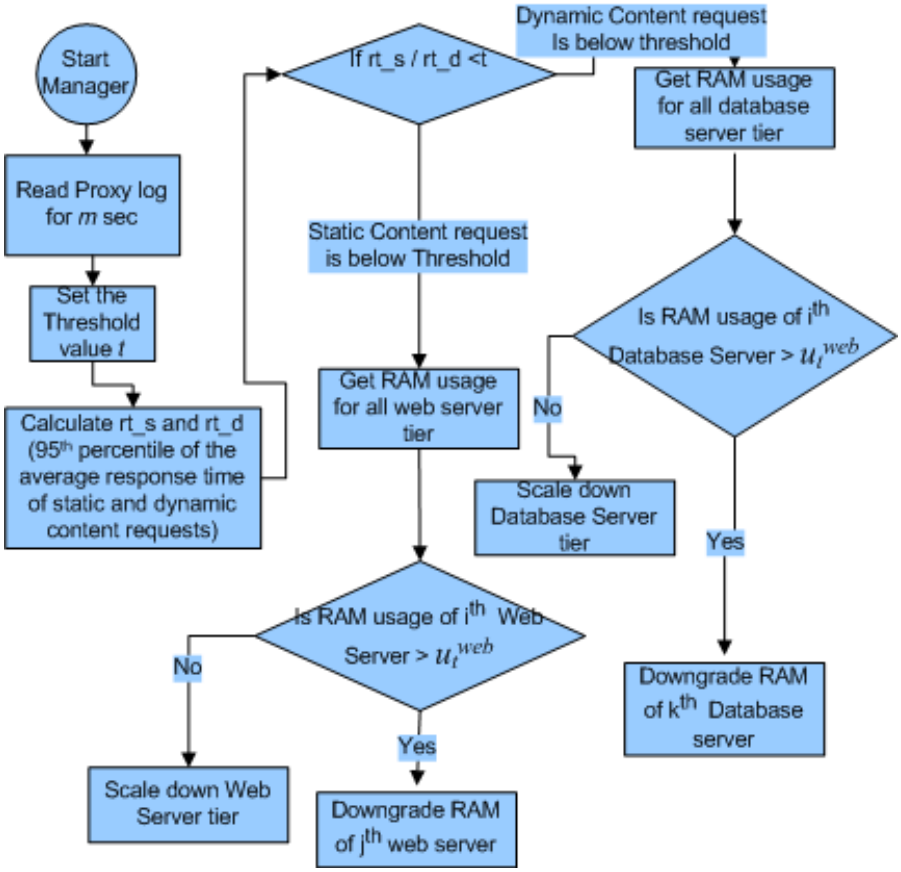

Figure 4.2: Model for Scaling down operation of Cloud Resources

4.4 Component diagram

The component diagram the various working components described for the deployment of the cloud. There are various components of the cloud. These are cloud controller, walrus, cluster controller, and nodes. The systematic component diagram is depicted in Figure 4.3.

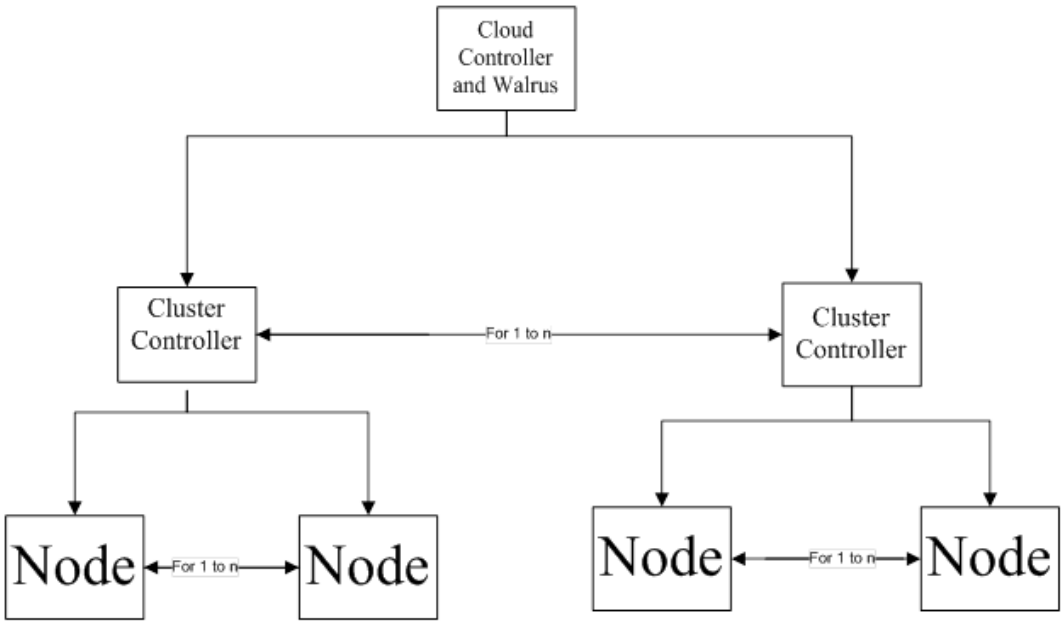

Figure 4.3: Component Diagram

The description of the various components of Figure 4.3 is described below:

1. Cloud Controller: It is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes high level scheduling decisions, and implements them by making requests to cluster controllers.
2. Cluster Controller: Gathers information about and schedules VM execution on specific node controllers, as well as manages virtual instance network
3. Node: Controls the execution, inspection, and terminating of VM instances on the host where it runs.
4. Role of Cloud controller: The CLC is a collection of web services which are best grouped by their roles into three categories:
    a. Resource Services perform system-wide arbitration of resource allocations, let users manipulate properties of the virtual machines and networks, and monitor both system components and virtual resources.
    b. Data Services govern persistent user and system data and provide for a configurable user environment for formulating resource allocation request properties.
    c. Interface Services present user-visible interfaces, handling authentication & protocol translation, and expose system management for providing tools

4.5 Sequence Diagram

The Figure 4.5 represents the sequence diagram for installation of the cloud as Infrastructure as a Service.

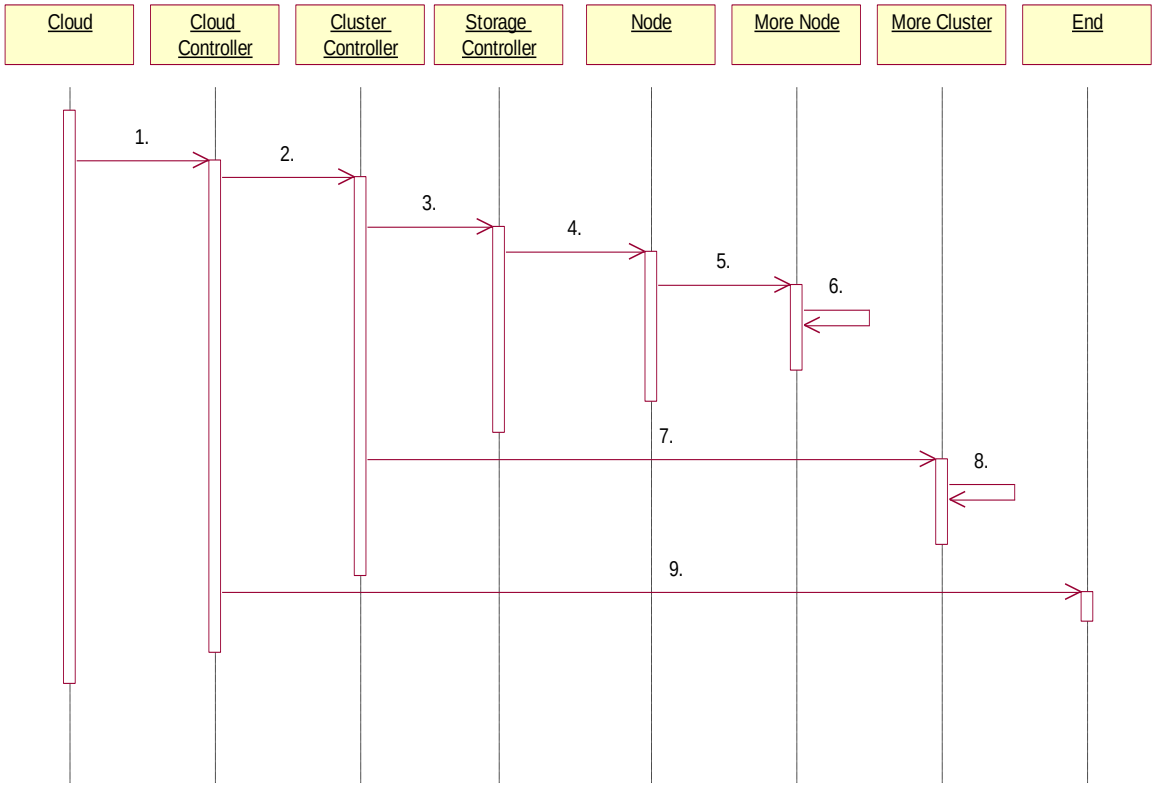

Figure 4.5: Sequence Diagram for installation of Cloud

The Cloud Controller, Cluster Controller, Storage Controller, Node and Clusters has been installed in the manner as depicted in Figure 4.5. As the need of the number of nodes will increase, more clusters will be needed to be installed on the cloud. The nodes of different requirement specifications will be installed on the cloud.

The Figure 4.6 represents the Sequence Diagram for the Working of the cloud as Infrastructure as a Service.

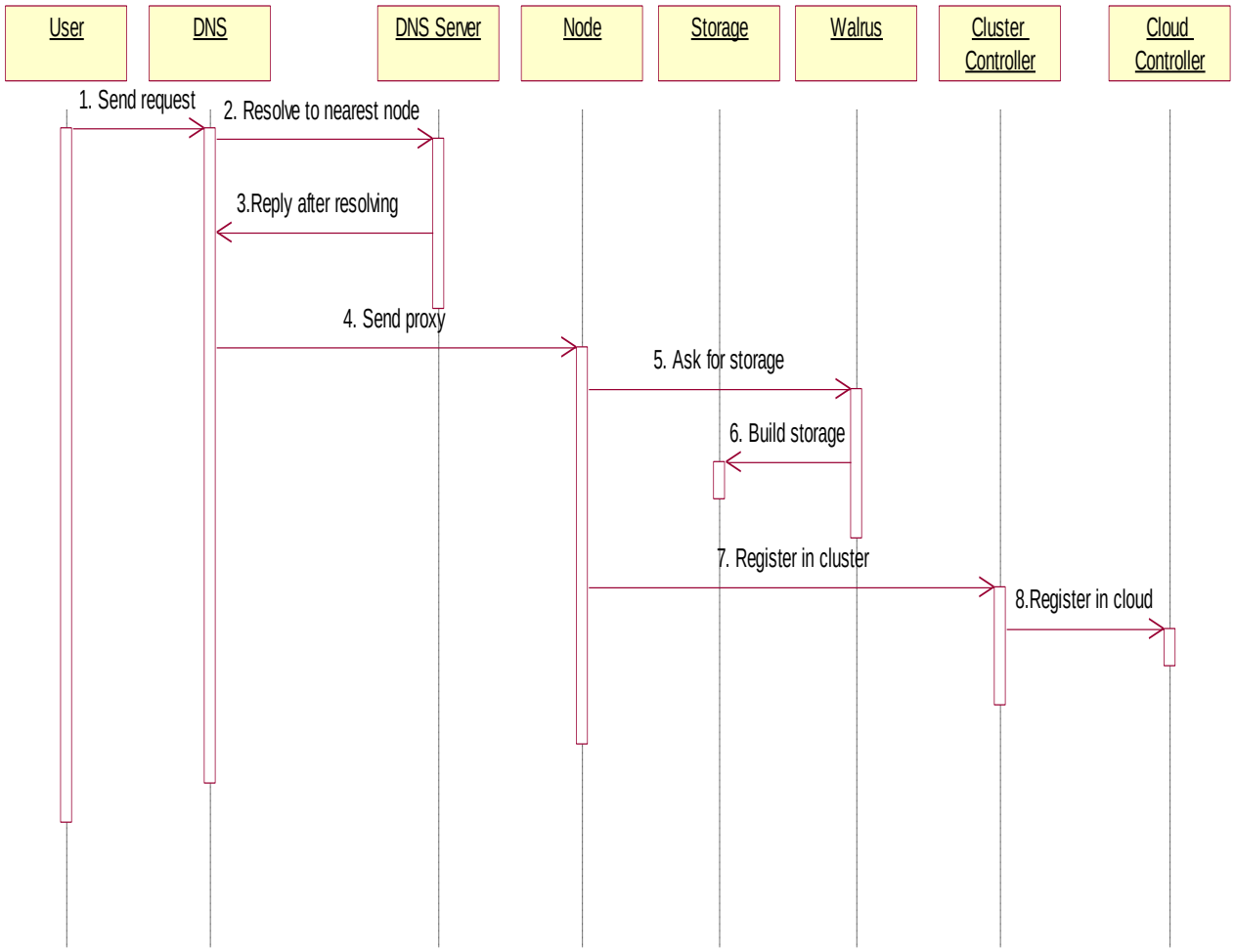

Figure 4.6: Sequence Model depicting the working of Cloud

The above diagram represents the sequence and manner in which the cloud resources will be deployed on the Infrastructure. The registration of the cloud, storage and cluster will be followed in the sequence depicted in the Figure 4.6. The installation of the cloud will be cloud controller. The cloud controller will manage various clusters. The cluster will contain different number of nodes on which the number of virtual machines will be running. Walrus acts as data storage for the cloud. It also acts as a VM image storage and service manager.

The above described prototype model is also needed to be validated in the cloud environment. The next chapter will focus on the deployment of the cloud environment and validation of the prototype.

Chapter 5

Experimental Results

In this section we describe the setup for an experimental evaluation of our proposed model based on a testbed cloud using the RUBiS Web application and a synthetic workload generator. The installation of cloud as IaaS has been done on Eucalyptus. Eucalyptus installation has been described in Appendix I.

5.1 Testbed Cloud

A small private heterogeneous compute cloud on two physical machines (Front-end, and Node1using EUCALYPTUS 3.2 is built. Figure 8 shows the design of our testbed cloud. Front-end as Intel ® Core 2 Quad machine with 2.66 GHz CPU. Node1 and Node2 is an Intel Core i5 with 2.53 GHz CPU. Front End, Node1 and Node2 have 4 GB RAM.

EUCALYPTUS set up is used to establish a cloud architecture comprising of one Cloud Controller (CLC), one Cluster Controller (CC), and two Node Controller (NCs). The CLC and CC is installed on a front-end node attached to both main LAN and the cloud's private network. The NCs is installed on two separate machines (Node1) and connected to the private network.

Table 5.1 describes the hardware specification of the EUCALYPTUS Cloud set up.

Table 5.1: Hardware Configuration of the physical machines used for setting up cloud

Node	Type	CPU	RAM
Front end	Intel Core 2 Quad	2.66 GHz	4 GB
Node1	Intel Core i5	2.53 GHz	4 GB
Node2	Intel Core i5	2.53 GHz	4 GB

The cloud installation is done by installing the CLC and CC on a front-end node attached to both our main LAN and the cloud's private network. The NCs are installed on two separate machines (Node1and Node2) connected to the private network. Each physical machine has the capacity to spawn a maximum number of virtual machines as shown (highlighted in Black) in the Figure 5.1, based on the number of cores.

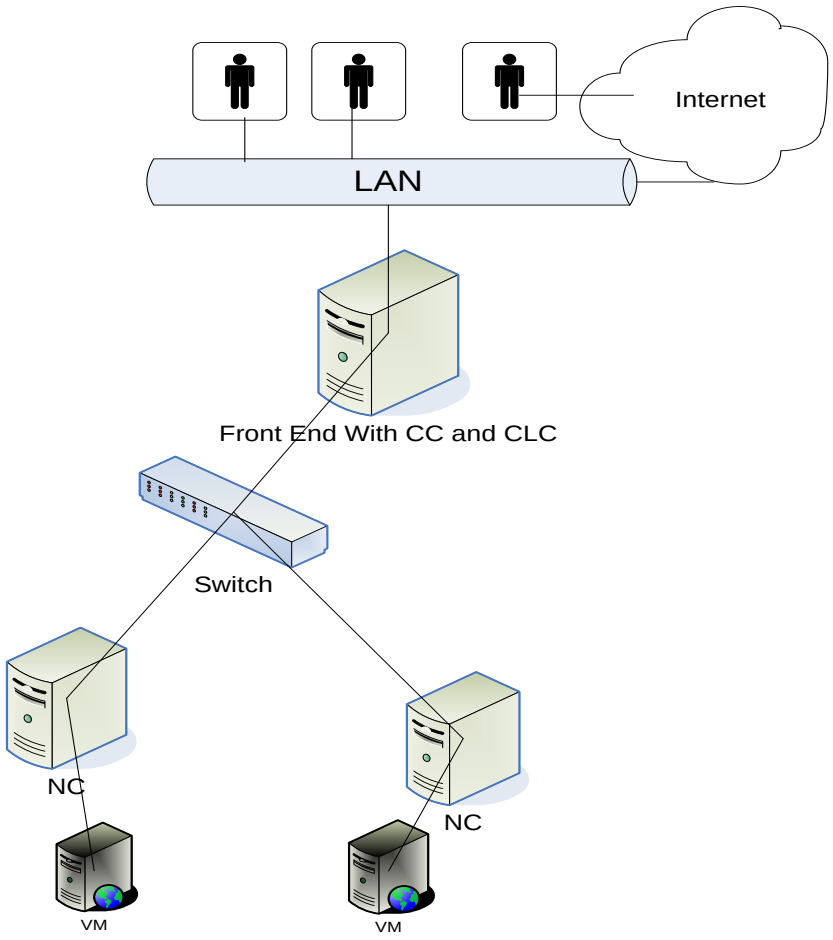

Figure 5.1: EUCALYPTUS-based testbed cloud using three physical machines.

5.2 Generation of Cloud Components

The various cloud components generated for the cloud environment are Cloud Controller, Cluster Controller, Walrus as Storage Controller and Nodes. They are registered on the Cloud as service components. The Figure 5.2 depicts the registrations of various components on Cloud.

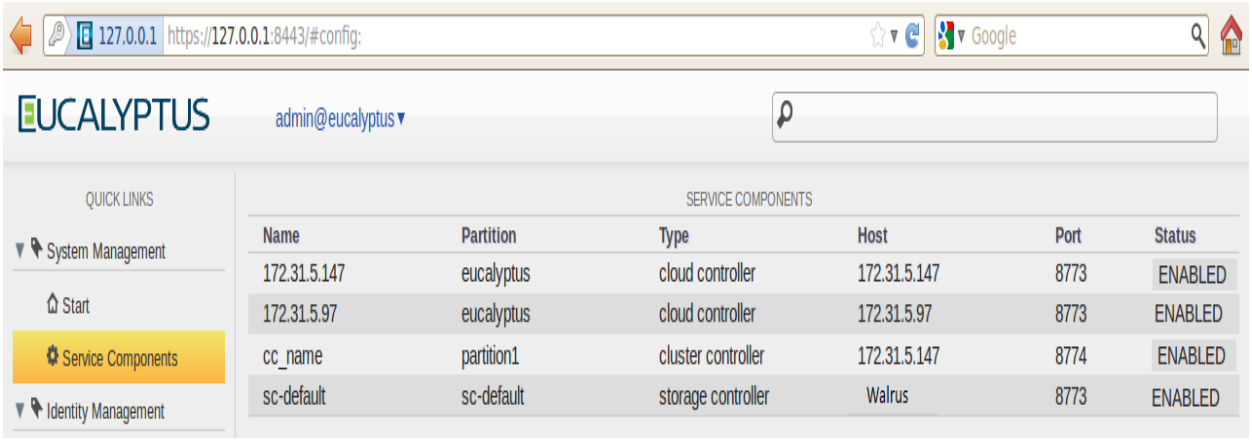

Figure 5.2: Cloud Components

5.3 Generation of Virtual machines Types

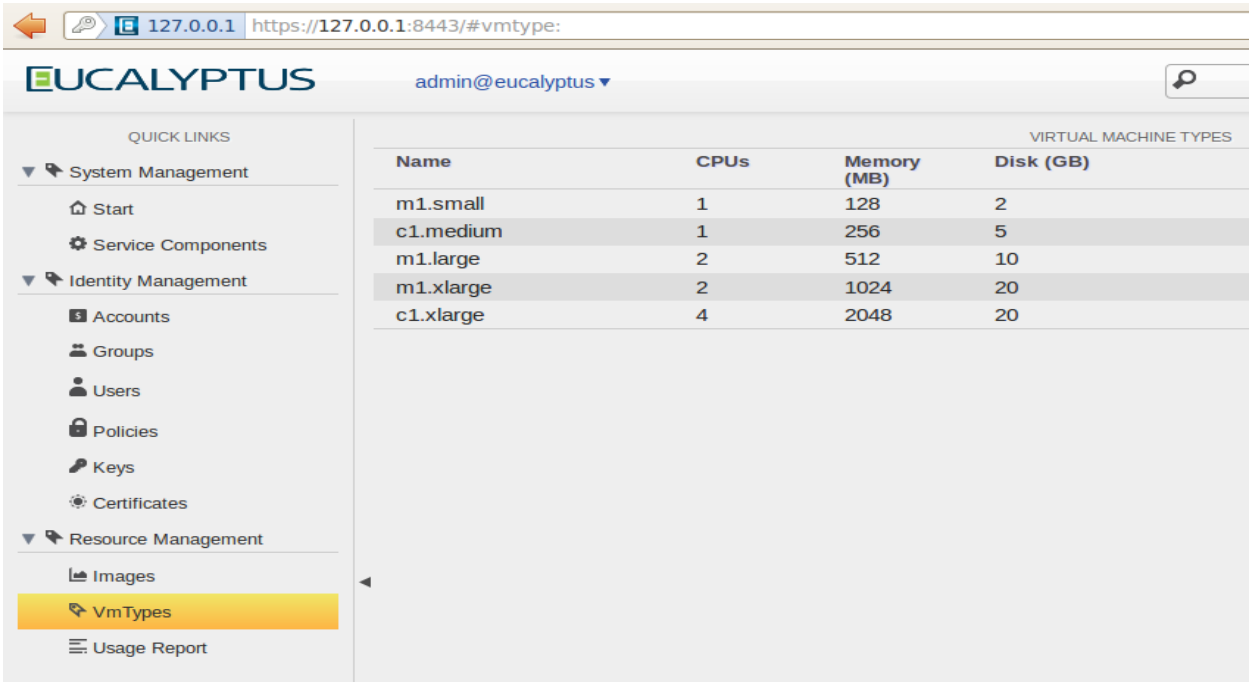

Figure 5.3: Virtual Machine Types

The various kind of the virtual machines types were; i) m1.small comprising of 1 core of CPU, 128 MB memory, 2 GB disk space; ii) c1.medium comprising of 1 core of CPU, 256 MB memory, 5 GB disk space; iii) m1.large comprising of 2 core of CPU, 512 MB memory, 10 GB disk space; iii) m1.xlarge comprising of 2 core of CPU, 1024 MB memory, 20 GB disk space; iv) m1.large comprising of 4 core of CPU, 2048 MB memory, 20 GB disk space. Figure 5.3 depicts the installation using Cloud.

5.4 Partitioning the machines

After these virtual machine types were created on the nodes, depending upon the specification of the nodes the number of virtual machine on each type were created on the node. The types can be described in the Figure 5.4.

```
root@FRONT:~# euca-describe-nodes
NODE    172.31.5.40     cc_name
root@FRONT:~# euca-describe-availability-zones verbose
AVAILABILITYZONE        partition1      172.31.5.147 arn:euca:eucalyptus:partiti
on1:cluster:cc_name/
AVAILABILITYZONE        |- vm types     free / max   cpu   ram  disk
AVAILABILITYZONE        |- m1.small     0004 / 0004   1    128     2
AVAILABILITYZONE        |- c1.medium    0004 / 0004   1    256     5
AVAILABILITYZONE        |- m1.large     0002 / 0002   2    512    10
AVAILABILITYZONE        |- m1.xlarge    0002 / 0002   2   1024    20
AVAILABILITYZONE        |- c1.xlarge    0001 / 0001   4   2048    20
```

Figure 5.4: Number of free and maximum virtual machine of each type

5.5 Bundling of the image

There are various types of image that can be run on the cloud. There is a need to bundle these image so that they can be registered into the walrus. It is necessary part to make the images run on the virtual machine. Two types of image were run on the cloud. First one was Ubuntu-9.04.x86.img and other was euca-centos-2011.07.02-i386.img. It is depicted in Figure 5.5.

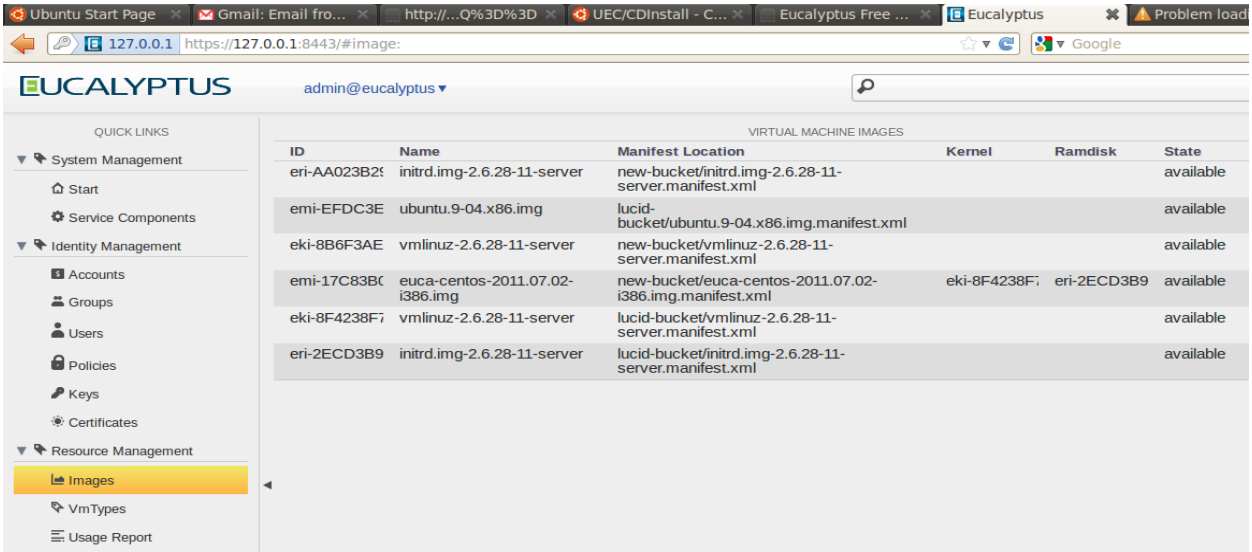

Figure 5.5: Bundling of Image

5.6 Running the Virtual Image

The desired virtual image is run on the virtual machine type on the node. For the experiment various types of images were run. After running the image the number of free and maximum machines of each type was reduced accordingly. Running of the virtual instance is depicted in Figure 5.6. The instance is in pending state because it is just starting up.

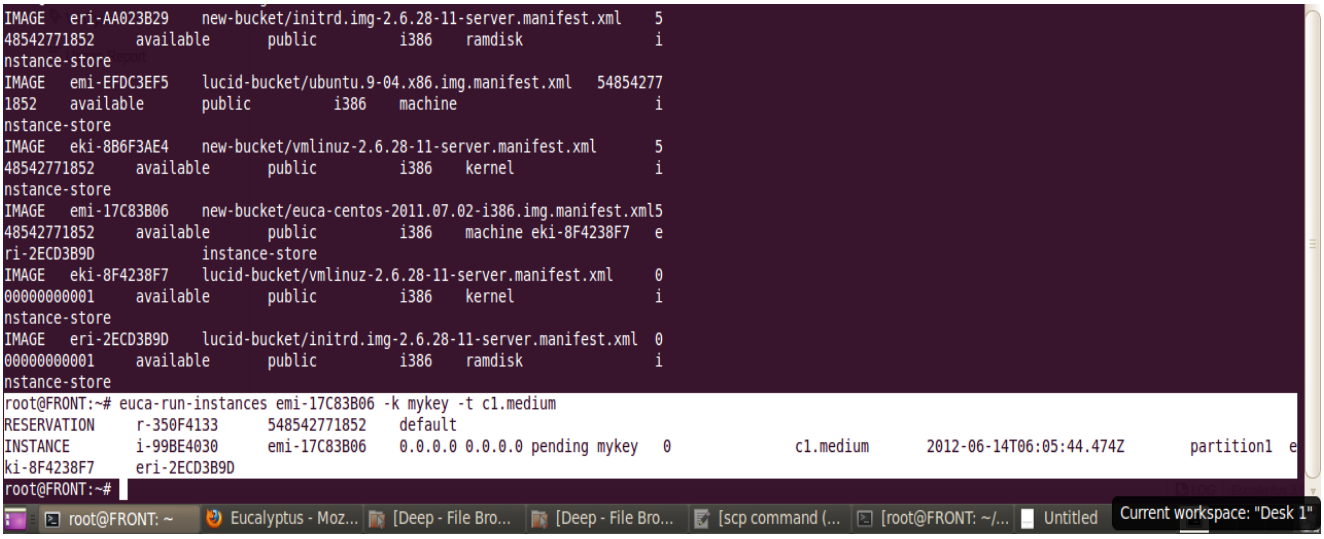

Figure 5.6: Running of the Virtual Instance

After some time the instance can come from pending state to running state. Figure 5.7 depicts that how the instance has come from pending state to the running state and it is being assigned an IP address on the Node. It is finally ready for the SSH operation.

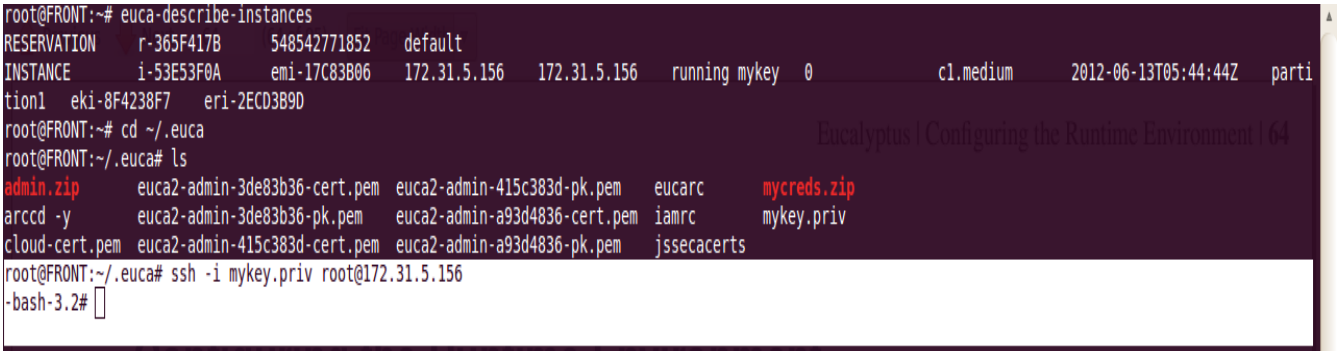

Figure 5.7: Running of Virtual Machine

The number of virtual instances will also decrease. Figure 5.8 depicts that how the number virtual instances decreases after the image i-53E53F0A is successfully run.

```
root@FRONT:~# service eucalyptus-cloud start
 * Starting Eucalyptus services                                    [ OK ]
root@FRONT:~# service eucalyptus-cc start
 * Starting Eucalyptus cluster controller eucalyptus-cc            [ OK ]
root@FRONT:~# service eucalyptus-cc cleanrestart
 * Restarting Eucalyptus cluster controller                        [ OK ]
root@FRONT:~# euca-describe-availability-zones verbose
AVAILABILITYZONE        partition1      172.31.5.147 arn:euca:eucalyptus:partiti
on1:cluster:cc_name/
AVAILABILITYZONE        |- vm types     free / max   cpu   ram  disk
AVAILABILITYZONE        |- m1.small     0004 / 0004   1    128     2
AVAILABILITYZONE        |- c1.medium    0004 / 0004   1    256     5
AVAILABILITYZONE        |- m1.large     0002 / 0002   2    512    10
AVAILABILITYZONE        |- m1.xlarge    0002 / 0002   2   1024    20
AVAILABILITYZONE        |- c1.xlarge    0001 / 0001   4   2048    20
root@FRONT:~# euca-describe-availability-zones verbose
AVAILABILITYZONE        partition1      172.31.5.147 arn:euca:eucalyptus:partiti
on1:cluster:cc_name/
AVAILABILITYZONE        |- vm types     free / max   cpu   ram  disk
AVAILABILITYZONE        |- m1.small     0003 / 0004   1    128     2
AVAILABILITYZONE        |- c1.medium    0003 / 0004   1    256     5
AVAILABILITYZONE        |- m1.large     0001 / 0002   2    512    10
AVAILABILITYZONE        |- m1.xlarge    0001 / 0002   2   1024    20
AVAILABILITYZONE        |- c1.xlarge    0000 / 0001   4   2048    20
root@FRONT:~#
```

Figure 5.8: Decrease in number of free VM instances

It is clearly observed that the number of VM instances is decreasing before and after the VM c1.medium was run on the Node. This is due to the reason that as 1 of the core of CPU gets occupied which leads to the decrease of the free instances of the cloud while the max instances will remain the same. The Figure 5.8 describes the instances that are free and max.

After doing ssh on the node JDK, apache7, sysstat, mysql server are installed on the virtual machine that will help in the deployment of the RUBiS [52]. The RUBiS will help in the generation of the synthetic workload of traffic on the machine. The workload instances can accordingly be analyzed. Figure 5.9 depicts the deployment of the apache and JDK on the virtual machine.

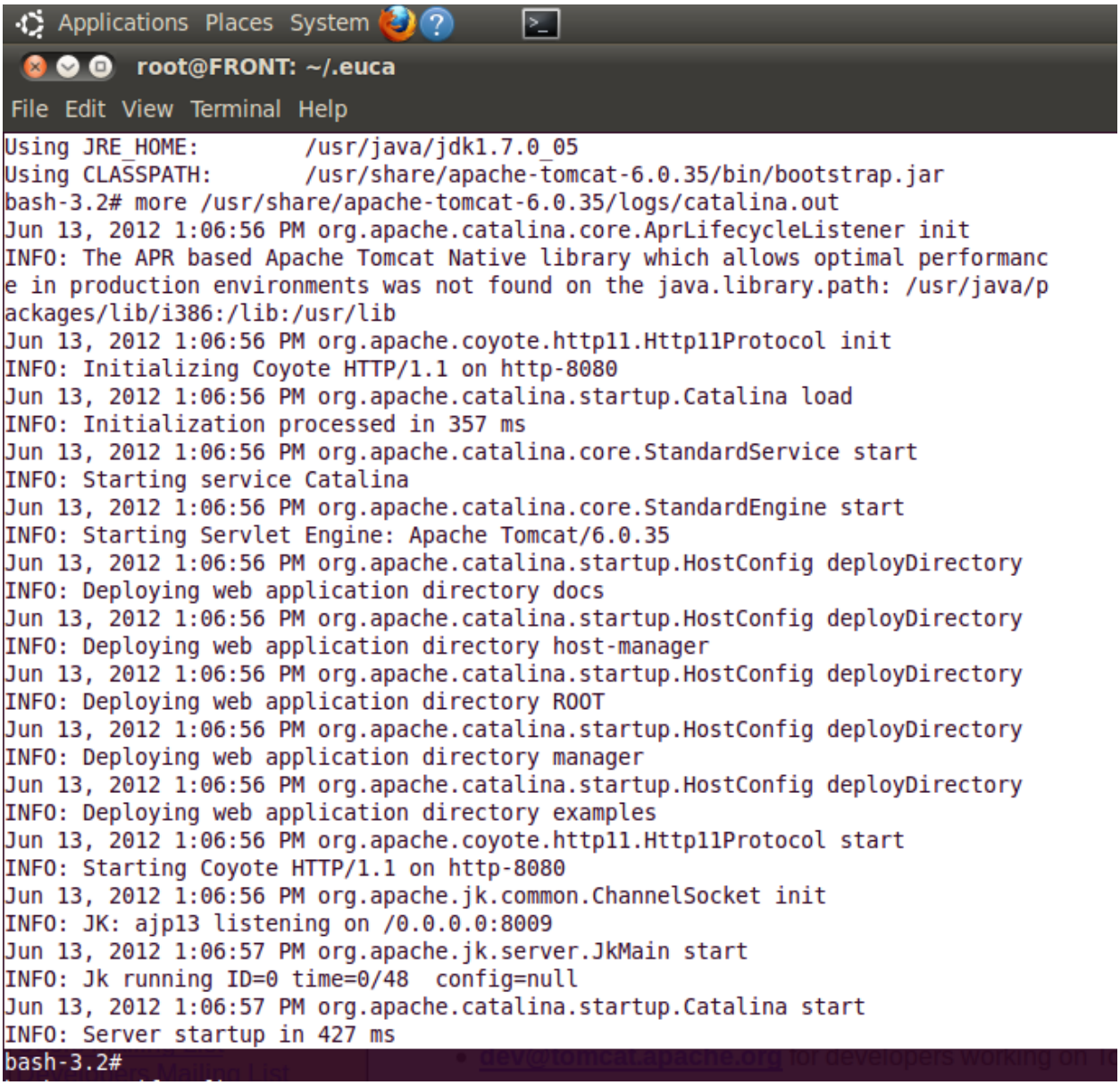

Figure 5.9: Installation of Java and Apache on Virtual Machine created on Node

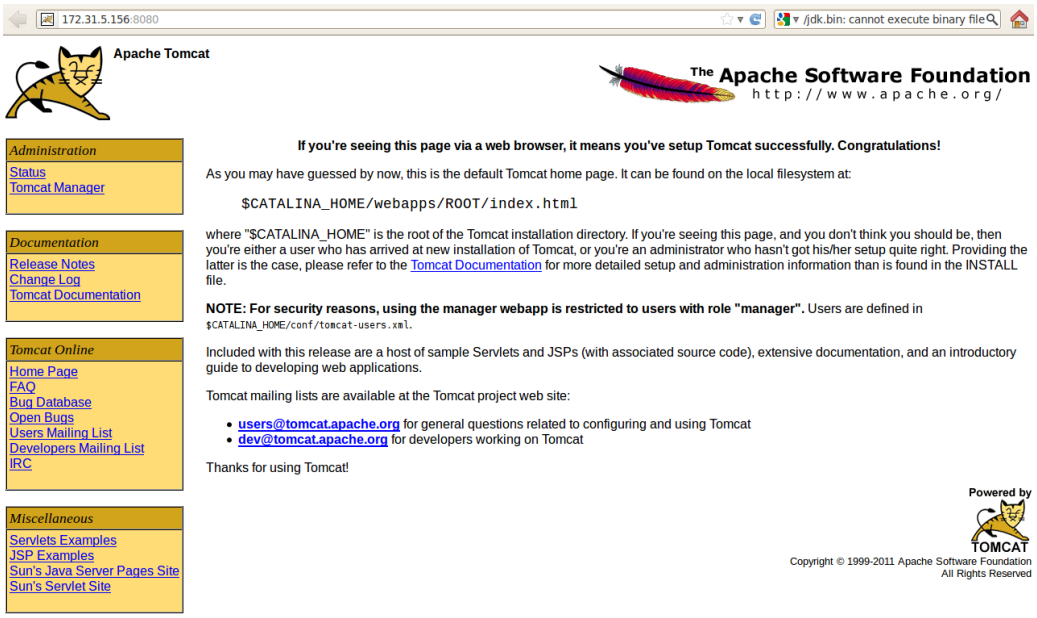

Figure 5.10: Apache Installed

Figure 5.10 depicts that the Apache that was successfully deployed on the virtual machine. The IP address 172.31.5.156:8080 is the IP address of the virtual machine. The instance was successfully deployed on the virtual machine.

5.7 Deployment of a Benchmark Application RUBiS

The .war file of RUBiS [56] which is web application for selling and purchase of the items, is deployed on the virtual machine. Following Figure 5.11 to Figure 5.15 depicts its deployment. It consisted of the registration of the RUBiS followed by the various steps for the bidding of the product.

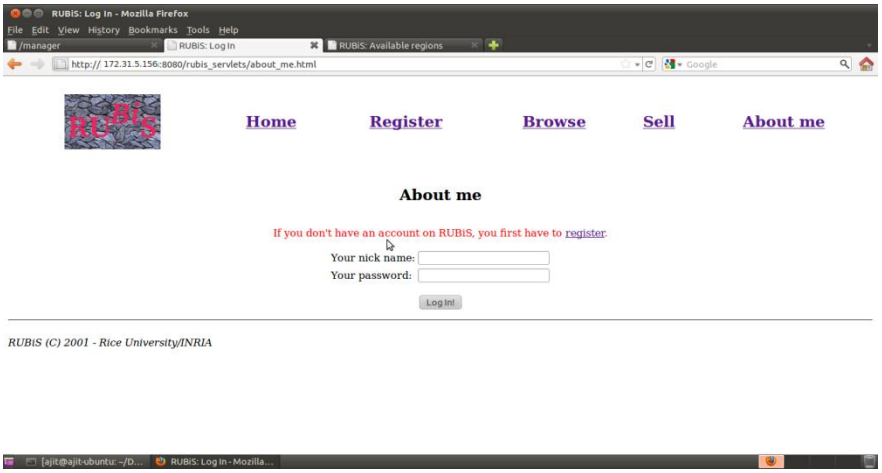

Figure 5.11: Home Page to login

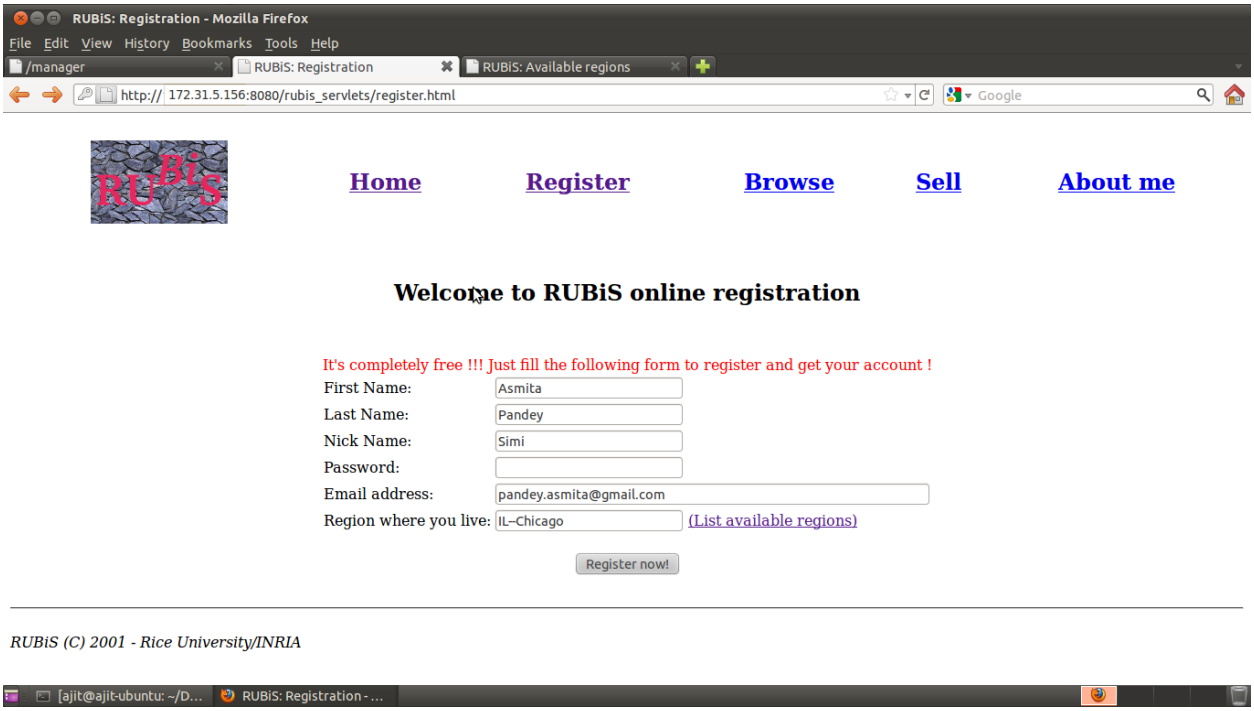

Figure 5.12: Registration page

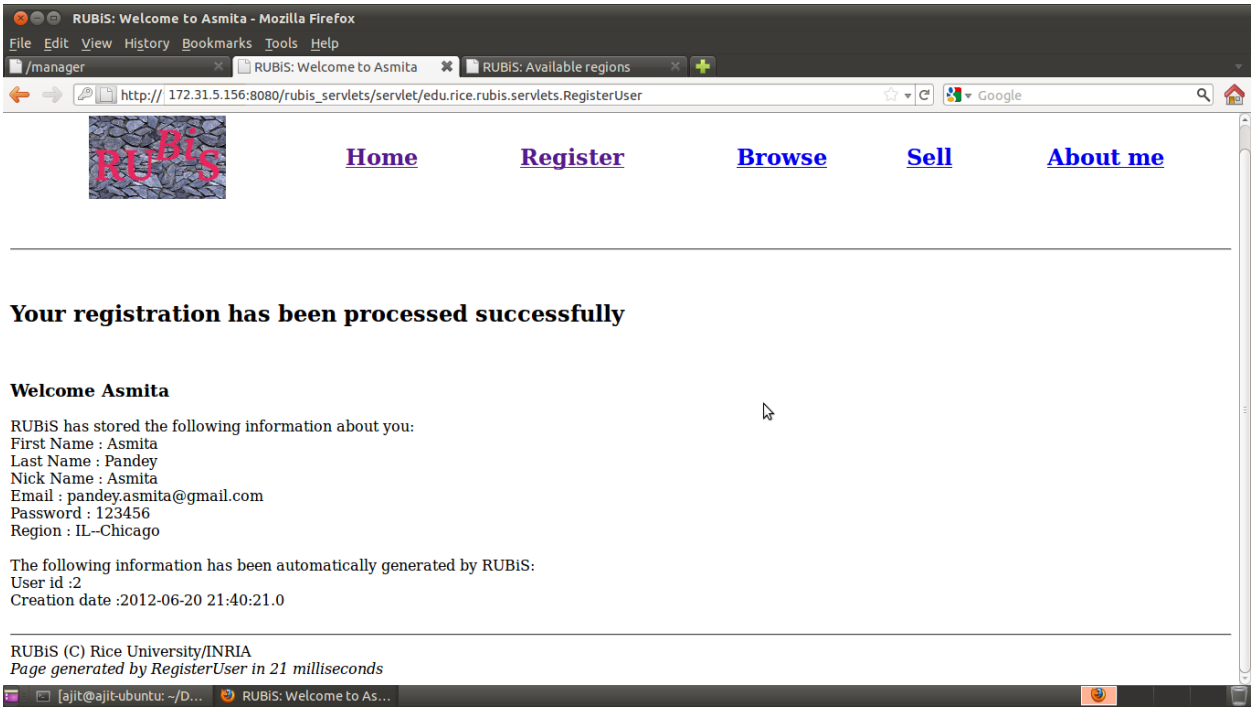

Figure 5.13: Registration Successful page

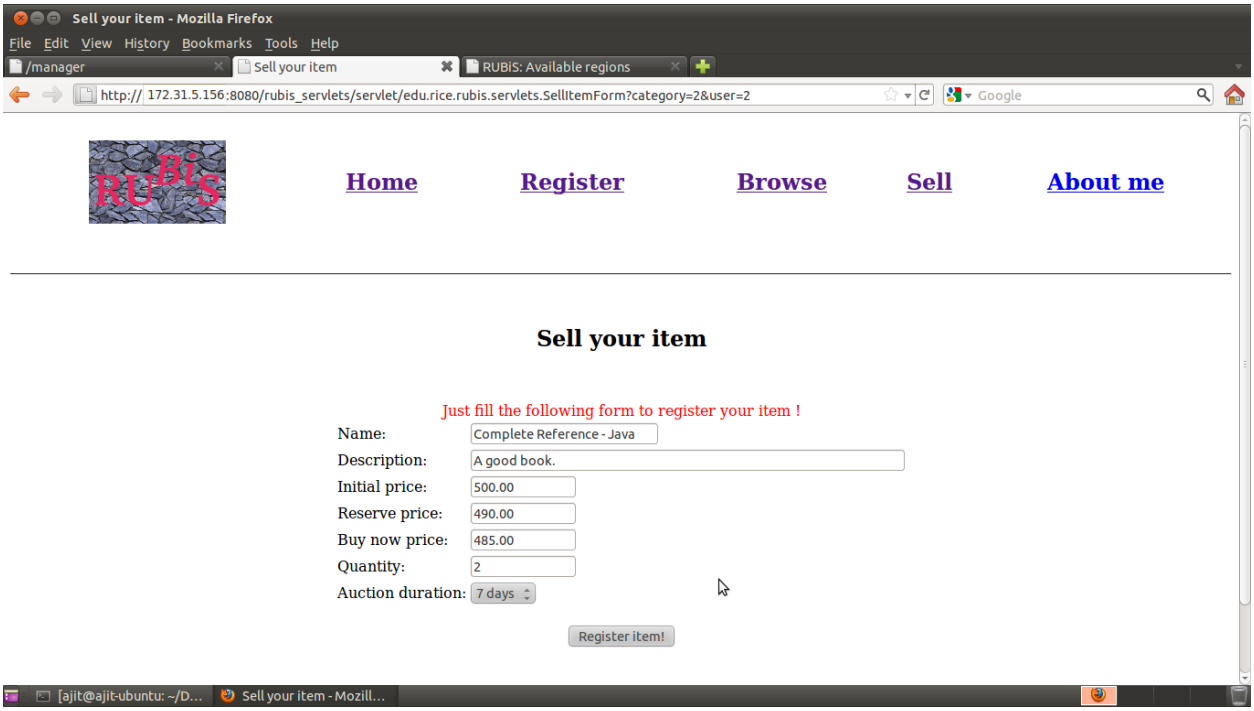

Figure 5.14: Selling of an Item page

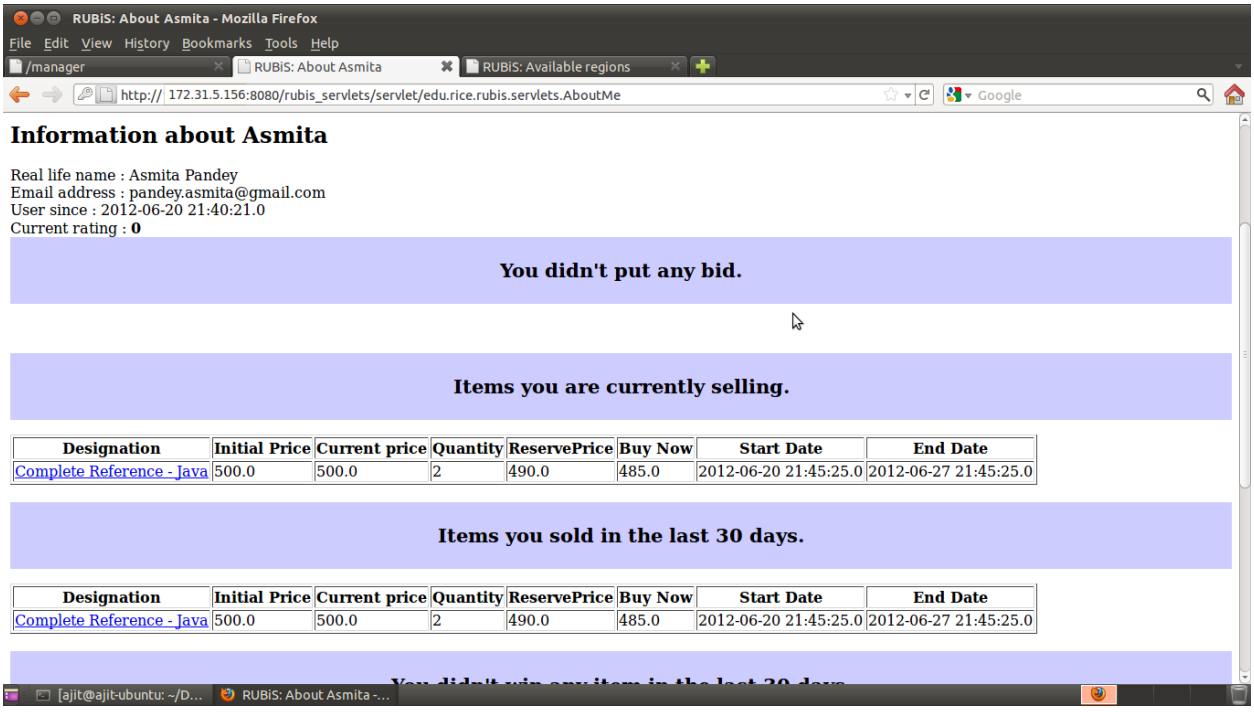

Figure 5.15: Status of the Item page

This application along with *httperf* [57] can be used for the synthetic workload generation on the RUBiS. This will lead to the increase in the traffic on the website by increasing the number of users per second. A user session emulates a visitor that browses items up for auction in specific

categories and geographical regions and also bids on items up for auction. As the web traffic will increase the number of web server tier and database server tier demand will also increase. This will also lead to the increase in the demand of the memory associated with each tier. Similarly as the web traffic will decrease the number of database tier and web server tier will decrease and this will lead to the decrease in the number of the memory associated with each tier.

5.8 Result and discussion.

The model was validated by generating traffic on RUBiS with the help of *httperf* which is a synthetic workload generator. Ten virtual machines were created for validating the work. On these virtual machines RUBiS application was deployed. These machines were accessed by another machine where the *httperf* synthetic workload generator was present which generated traffic over it. The web server tier and database server tier were collectively installed on the single node.

The graph in Figure 5.16 clearly depicts that how the response time varies when the number of request increases. This machine has configuration of 1 core with 2.53 GHz processor where number of request is 10,000.

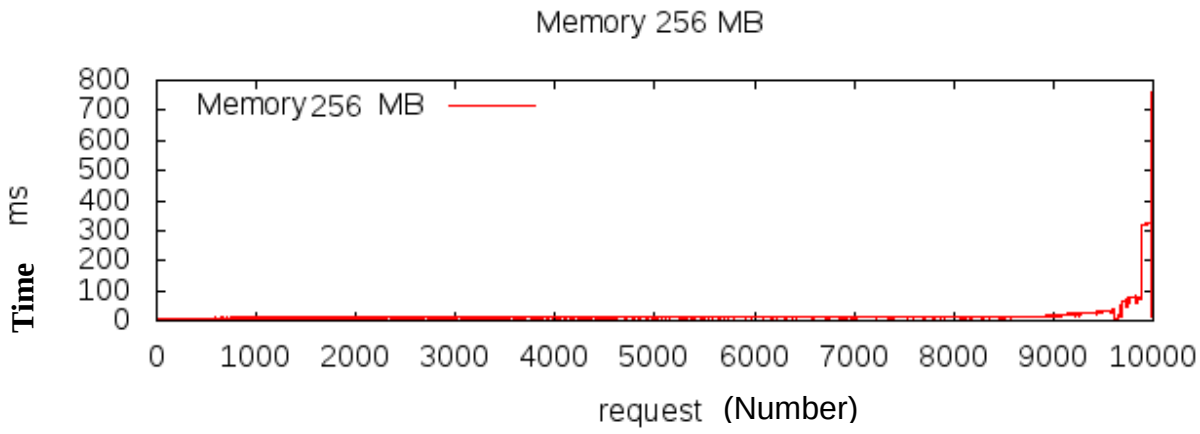

Figure 5.16: Response Time variation on VM with 1 core, 256 MB Memory and traffic load of 10000 requests

The graph in Figure 5.17 clearly depicts that how the response time varies when the number of request increases. This machine has configuration of 1 core with 2.53 GHz processor where number of request is 100,000.

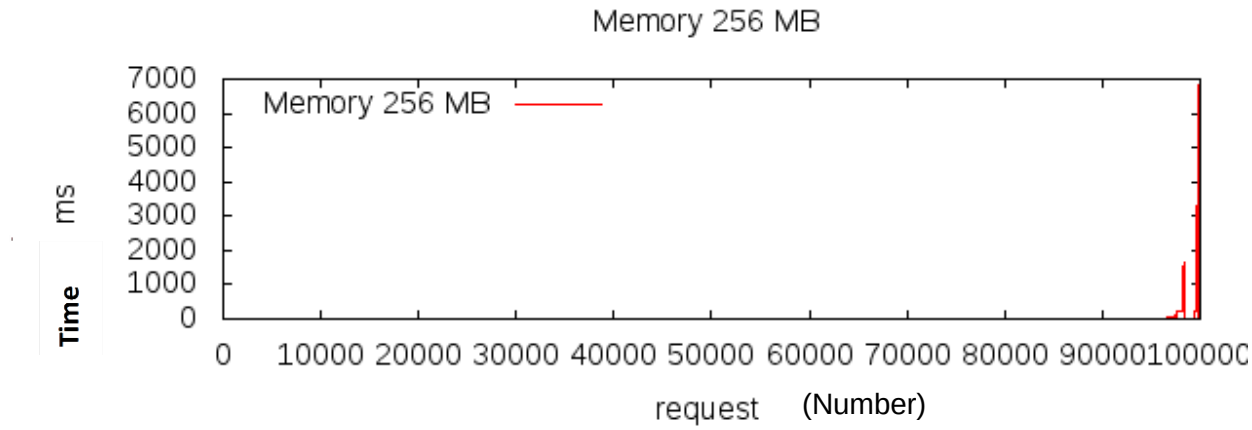

Figure 5.17: Response Time variation on VM with 1 core, 256 MB Memory and traffic load of 100,000 requests

The graph in Figure 5.18 clearly depicts that how the response time varies when the number of request increases. This machine has configuration of 1 core with 2.53 GHz processor where number of request is 100,000.

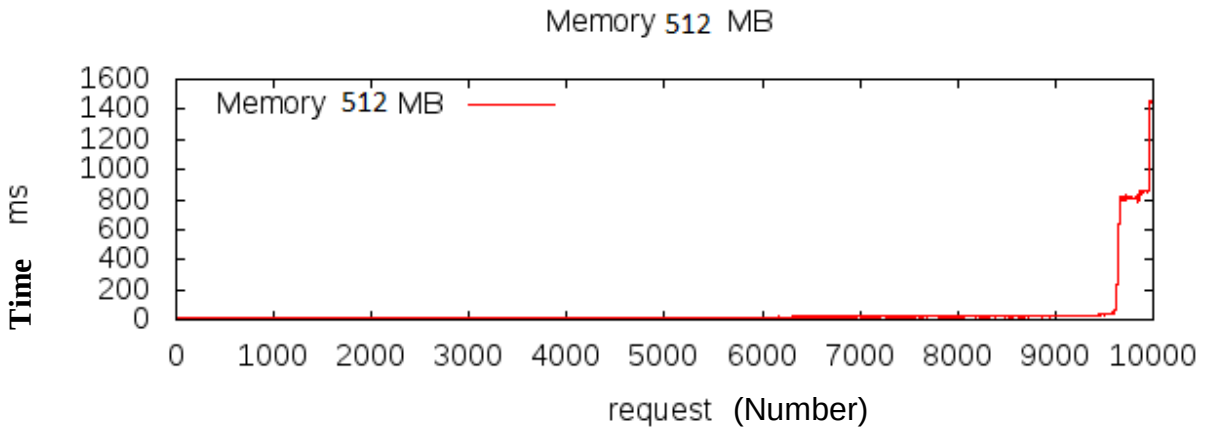

Figure 5.18: Response Time variation on VM with 1 core, 512 MB Memory and traffic load of 10,000 requests

The graph in Figure 5.19 clearly depicts that how the response time varies when the number of request increases. This machine has configuration of 1 core with 2.53 GHz processor where number of request is 100,000.

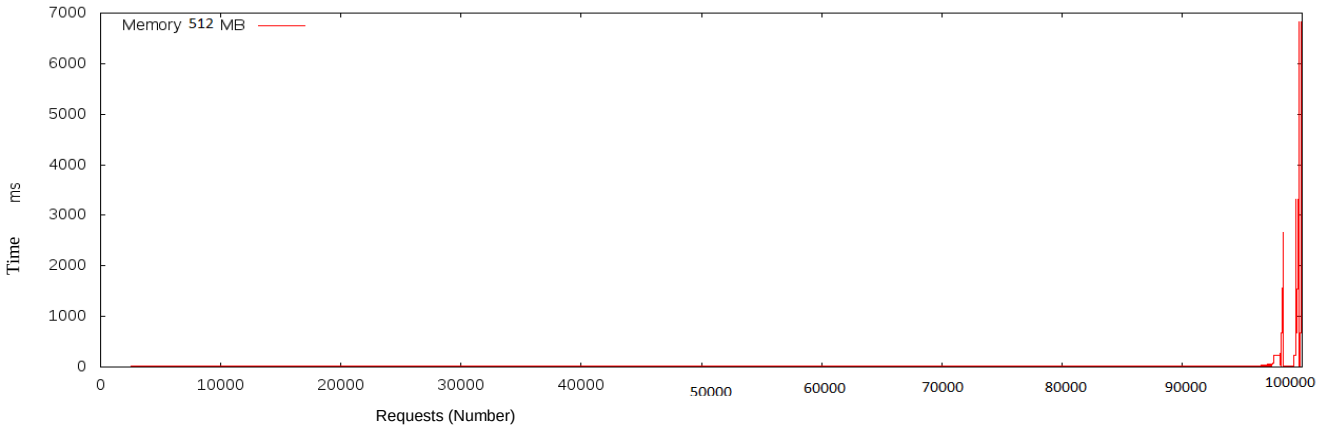

Figure 5.19: Response Time variation on VM with 1 core, 512 MB Memory and traffic load of 100,000 requests

The graph in Figure 5.20 clearly depicts that how the response time varies when the number of request increases. This machine has configuration of 1 core with 2.53 GHz processor where number of request is 10,000.

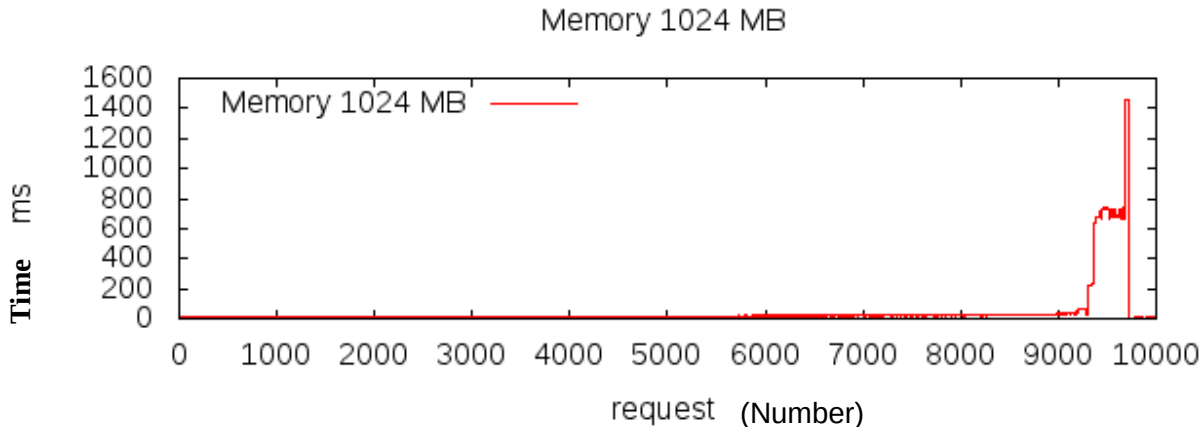

Figure 5.20: Response Time variation on VM with 1 core, 1024 MB Memory and traffic load of 10,000 requests

The graph in Figure 5.21 clearly depicts that how the response time varies when the number of request increases. This machine has configuration of 1 core with 2.53 GHz processor where number of request is 100,000.

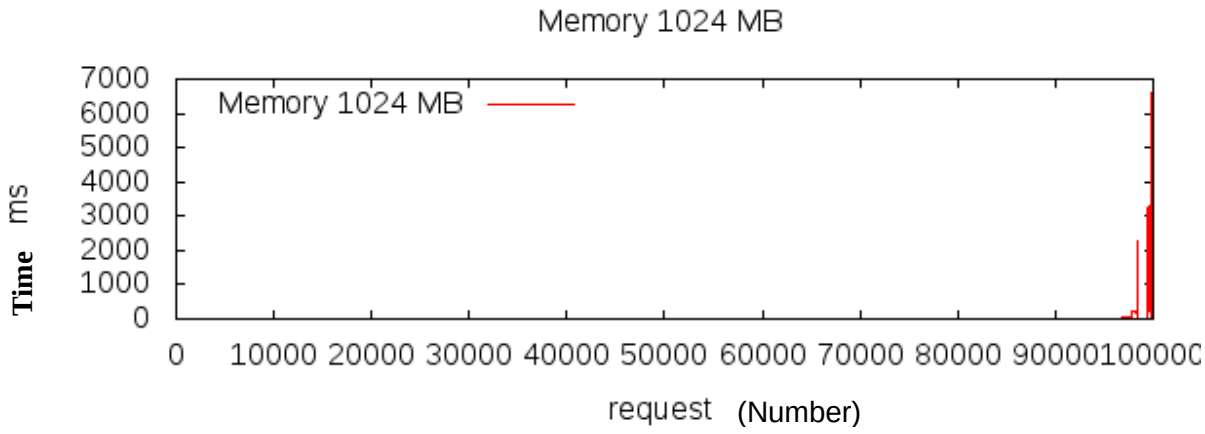

Figure 5.21: Response Time variation on VM with 1 core, 1024 MB Memory and traffic load of 100,000 requests

The data from the collected statistics was analyzed with different aspects and various graphs were plotted on the basis of the analysis of the graph.

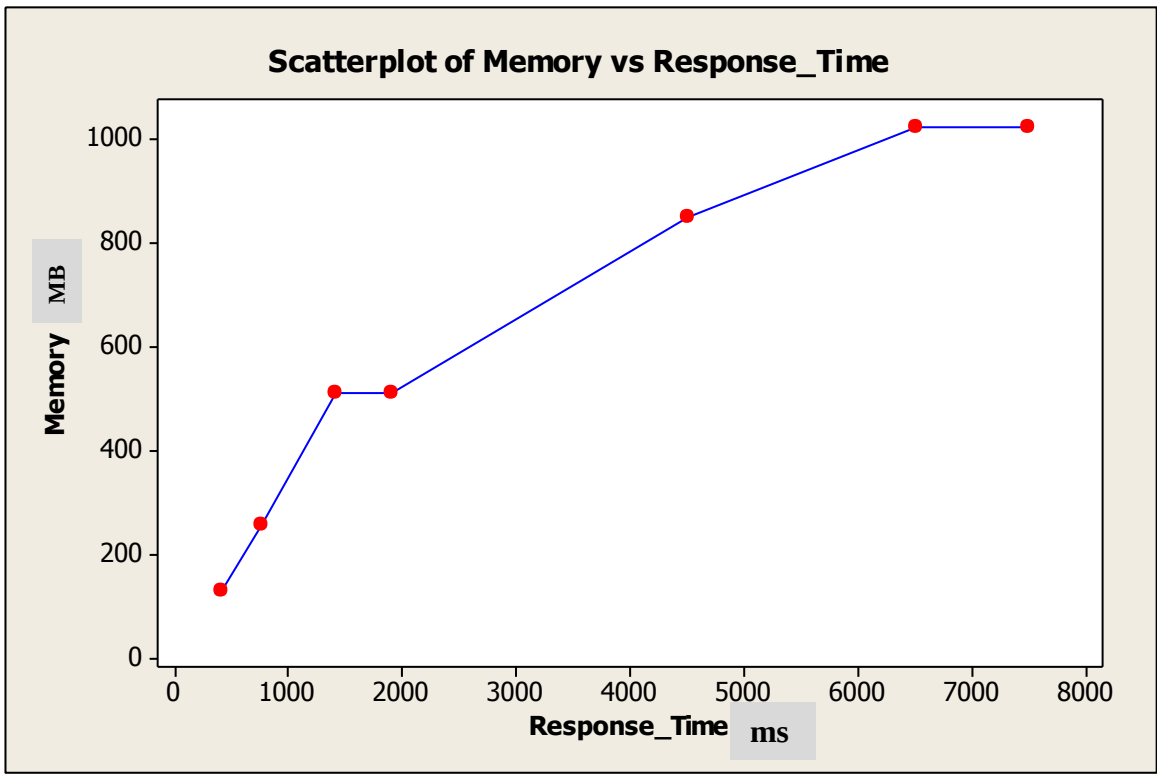

Figure 5.22: Memory with Response Time Graph

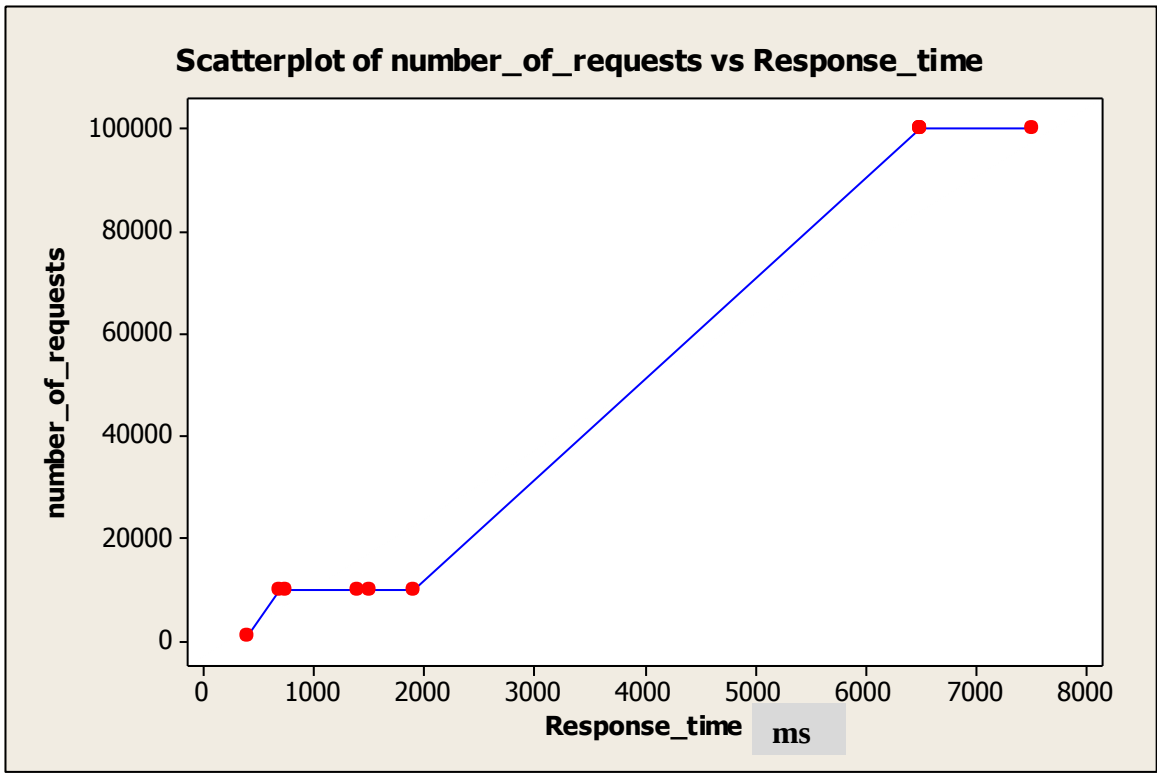

Figure 5.23: Number of requests with Response Time Graph

Figure 5.22 clearly depicts that the graph is an equation of polynomial equation of degree 2. Thus the predictive model described in the equation 3 and 4 with sufficient number of statistics has been validated. The graph corresponds to the curve of degree two. Thus the regression model proposed as Memory Predictive model in section 4.1 has been validated here. The Equation 3 and Equation 4 depicted there correspond to equation of the curve, hence is been experimentally proven in the Figure 5.23 which shows a graph having curve of degree two.

The next chapter concludes the thesis.

Chapter 6

Conclusion & Future Scope

This chapter discusses the conclusion of the work presented in the thesis. The chapter ends with a discussion of the future direction and scope of research.

6.1 Thesis Contribution

a) This thesis compares the various resource provisioning techniques on the cloud. A comparative study of available techniques has also been done

b) A memory predictive model has been proposed and designed to provision multiple resources

c) EUCALYPTUS Cloud testbed has been installed on multiple physical nodes.

d) A benchmark application RUBiS has been deployed on EUCALYPTUS testbed.

e) The proposed memory predictive model has been validated using EUCALYPTUS testbed

6.2 Future Scope

a) The proposed model can be integrated with various resource prediction techniques for the management of the adequate amount of resources on the cloud for optimized resource management.

b) In order to further reduce the complexity the code optimization technique can be implemented to reduce response time.

c) The proposed model can be further enhanced and validated on the basis of larger statistical data.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the Clouds: A Berkeley View of Cloud computing" in Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

[2] I. Foster, Y. Zhao, I. Raicu, S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared" in Grid Computing Environments Workshop, 2008.

[3] L. Youseff, M Butrico and D. D. Silva, "Toward a Unified Ontology of Cloud Computing", in Grid Computing Environments Workshop (GCE 2008), Austin, Texas, USA, November 2008, pp. 1–10 (2008).

[4] M. P. Rad, A. S. Badashian, G. Meydanipour, M. A. Delcheh, M. Alipour, and H. Afzali, " A survey of cloud platforms and their future" in Lecture Notes of Computer Science, 5592, pp. 788-796 (2009).

[5] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2, 2011.

[6] D. Johnson, K. Murari, M. Raju, R. B. Suseendran and Y Girikumar "Eucalyptus Beginner's Guide – UEC Edition". Published by CSS Corp. Pvt. Ltd, v1.0, 25 May 2010.

[7] G. Singh, C. Kesselman E. Deelman. "Performance Impact of Resource Provisioning on Workflows", in USC ISI Technical Report, 2006.

[8] A. Bechtolsheim, "Cloud Computing". Available at http://netseminar.stanford.edu/seminars/Cloud.pdf.

[9] Google App Engine. https://developers.google.com/appengine, [Last accessed on May 2012].

[10] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the Clouds: A Berkeley View of Cloud computing" in Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

[11] P. D. Kaur, I. Chana, "Unfolding the Distributed Computing Paradigms" in International Conference on Advances in Computer Engineering, 2010.

[12] M. Papazoglou and D. Georgakopoulos, "Service-Oriented Computing" in Communications of the ACM, 46(10):2528, 2003.

[13] X. Wang, Z. Dua, Y. Chen, and S. Li, "Virtualization-based autonomic resource management for multi-tier Web applications in shared data center" in The Journal of Systems and Software, 81. 2008, pp 1591–1608.

[14] B. Urgaonkar and P. Shenoy. "Sharc: Managing CPU and Network Bandwidth in Shared Clusters" in Technical Report TR01-08, Department of Computer Science, University of Massachusetts, October, 2001.

[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "Eucalyptus opensource cloud-computing system" in CCA08: Cloud Computing and Its Applications, 2008.

[16] G. Chen, A. Gillen, "KVM for Server Virtualization: An Open Source Solution Comes of Age Sponsored by: IBM" in WHITE PAPER, October 2011.

[17] D. Milojicic, I. M. Llorente, and R. S. Montero, "Opennebula: A cloud management tool" in. IEEE Internet Computing, vol 15, pp 11–14, 2011

[18] K. Keahey, "Cloud Computing with Nimbus". Available at http://www.nimbusproject.org/files/nimbus-xtreemOS_Sept2009.pdf

[19] C. Baun, M. Kunze, J. Nimis and S Tai, "Chapter 6 Open Source Cloud Stack", Published by Springer-Verlag Berlin Heidelberg, pp 49-62, 2011

[20] K Pepple, "Deploying OpenStack" Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, July 2011.

[21] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic Provisioning of Multi-Tier Internet Applications" in 2nd International Conference on Autonomic Computing (ICAC-2005), Seattle,WA, pp 217-228, june 2005.

[22] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards Autonomic Workload Provisioning for Enterprise Grids and Clouds" in Proceedings of

the 10th IEEE International Conference on Grid Computing (Grid 2009), Banff, Alberta, Canada, October 13-15 (2009), pp 50-57, 2009.

[23] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, in "A View of Cloud Computing" in Communications of the ACM, 53,4, pp. 50-58, April 2010.

[24] G. Goth, "Virtualization: Old technology offers huge new potential," in IEEE Distributed Systems Online, vol. 8, no. 2, pp 1-4, Feb. 2007.

[25] J. Torres, D. Carrera, K. Hogan, R. Gavalda, V. Beltran and N. Poggi, "Reducing wasted resources to help achieve green data centers" in Proc. of IEEE International Symposium on Parallel and Distributed Processing,pp 1–8, Apr. 2008.

[26] G. Lawton, "Powering down the computing infrastructure" in Computer, Vol. 40, No 2, 2007, pp 16-19, 2007.

[27] G. Goth "Virtualization: Old technology offers huge new potential" in IEEE Distributed Systems Online, vol. 8, no. 2, pp 1-4, Feb. 2007.

[28] J. S. Turner, and D. E. Taylor, "Diversifying the internet" in Proc. of GLOBCOM'05, pp 755–760, Dec. 2005.

[29] P. Mell, and T. Glance, "The NIST definition of cloud computing" Available at: http://csrc.nist.gov/groups/SNS/cloud-computing/ [Last accessed May 18, 2012].

[30] Kahuna Technology Group Inc, "Why Software as a Service and Why Google Apps?". Available at http://www.ktgdenver.com/sites/default/files/KTG%20FAQ%20Why%20Google%20Apps_0.pdf

[31] Ron Zalkind, CTO CloudLock Inc, "Protecting Your Data in Google Docs Compliance in The Cloud" Available at http://173.193.230.213/PDF/protecting-your-data-in-google-docs.pdf

[32] Red Hat Reference Architecture Series, "Getting Started with Cloud Computing:Amazon EC2 on Red Hat Enterprise Linux". Available at

http://www.redhat.com/f/pdf/Getting_Started_with_Cloud_Computing_Amazon_EC2_on_Red_Hat_Enterprise_Linux.pdf

[33] The weather Channel, "The Weather Channel Brand Guidelines". Available at http://press.weather.com/downloads/TWCguide_vendor.pdf

[34] A. Zahariev "Google App Engine" TKK T-110.5190 Seminar on Internetworking 2009. Available at http://www.cse.hut.fi/en/publications/B/5/papers/1Zahariev_final.pdf

[35] Windows Azure Platform. Retrieved April 20, 2010, from Microsoft: http://www.microsoft.com/windowsazure/.

[36] C. Dwyer, S. R. Hiltz and K. Passerini, "Trust and Privacy Concern Within Social Networking Sites: A Comparison of Facebook and MySpace" in Proc AMCIS 2007.

[37] Amazon Elastic Compute Cloud User Guide API Version 2012-06-01 http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-ug.pdf

[38] P. Bodik, R. Griffith, C.Sutton, A. Fox, M. Jordan, and D. Patterson, M. Andreolini, S. Casolari, M. Colajanni and M. Messori, " Dynamic load management of virtual machines in a cloud architectures" in Proceedings of the First International Conference on Cloud Computing, 2009.

[39] G. Jung, K. Joshi and M. Hiltunen, "Performance aware regeneration in virtualized multitier applications" in PFARM '09: Proceedings of the Proactive Failure Avoidance Recovery and Maintenance, 2009.

[40] G. Jung, K. Joshi, M. Hiltunen, R. Schlichting, and C. Pu, "Generating adaptation policies for multi-tier applications in consolidated server environments" in Proc. 5th IEEE Intl. Conf. on Autonomic Computing, pp 23–32, June 2008.

[41] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Black-box and gray-box strategies for virtual machine migration" in USENIX NSDI'07: Proceedings of the Symposium on Networked Systems Design and Implementation, pp 229–242, 2007.

[42] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. Jordan and D. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters", HotCloud'09, 2009.

[43] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications" in ACM SIGMETRICS international conference on Measurement and modeling of computer systems, vol. 33. ACM, pp 291–302, 2005.

[44] D. Villela, P. Pradhan and D. Rubenstein, "Provisioning servers in the application tier for e-commerce systems" in Proceedings of the 12th IWQoS, pp 57-66, June 2004.

[45] A. Dubey, R. Mehrotra, S. Abdelwahed and A. Tantawi, "Performance modeling of distributed multi-tier enterprise systems" in MAMA '09: Proceedings of the Performance Modeling of Distributed Multi-Tier Enterprise Systems, Eleventh Workshop on Mathematical Performance Modeling and Analysis, 2009.

[46] The Apache Software Foundation, "Apache DayTrader Benchmark," 2005, http://cwiki.apache.org/GMOxDOC20/daytrader.html

[47] G. Reig, J. Alonso and J. Guitart, "Deadline Constrained Prediction of Job Resource Requirements to Manage High-Level SLAs for SaaS Cloud Providers" in Dept. d'Arquitectura de Computadors, Universitat Polit`ecnica de Catalunya, Barcelona, Spain, Tech. Rep. UPC-DAC-RR-2010-9, April 2010.

[48] G. Reig, J. Alonso and J. Guitart, "Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud" in 9th IEEE International Symposium on Network Computing and Applications (NCA'10), 162-167, 2010.

[49] G. Jungy, K. R. Joshiz, M. A. Hiltunenz, R. D. Schlichtingz, and C. Puy, "Generating adaptation policies for multi-tier applications in consolidated server environments" in ICAC '08: Proceedings of the International Conference on Autonomic Computing, pp 291–302, 2008.

[50] W. Iqbal, M. Dailey and D. Carrera, "SLA-Driven Adaptive Resource Management for Web Applications on a Heterogeneous Compute Cloud" in CloudCom Springer-Verlag Berlin Heidelberg, LNCS 5931, pp 243–253, 2009.

[51] Z. Gong, X. Gu and J. Wilkes, 2010. PRESS: PRedictive Elastic ReSource Scaling for Cloud Systems. Proc. CNSM, 9-16, 2010.

[52] W. Iqbal, M. Dailey and D Carrera, "SLA-Driven Dynamic Resource Management for Multi-tier Web Application in a Cloud" in IEEE'10: International Conference on Cluster, Cloud and Grid Computing, pp 832-837, 2010.

[53] W. Iqbal, M. Dailey, D Carrera and P. Janecek, "SLA-Driven Autonomic Bottleneck Detection and Resolution for Read Intensive Multi-tier Applications Hosted on a Cloud" in Springer-Verlag Berlin Heidelberg, LNCS 6104, pp 37-46, 2010.

[54] W. Iqbal, M. Dailey, D Carrera and P. Janecek, "Adaptive Resource Provisioning for Read Intensive Multi-tier Applications in the Cloud" in Future Generation Computer Systems, vol 27 (6), pp 871–879, 2011.

[55] W. Iqbal, M. Dailey, D Carrera and P. Janecek, "Adaptive Resource Provisioning on Heterogeneous Compute Cloud" in Future Generation Computer Systems, vol 27 (6), pp 871–879, 2011

[56] RUBiS: http://rubis.ow2.org/download/rubisva_v1.0.pdf

[57] D. Mosberger, T. Jin, httperf - a tool for measuring web server performance, in: In First Workshop on Internet Server Performance, ACM, pp 59-67, 1998.

LIST OF PUBLICATIONS

Asmita Pandey, Dr. Inderveer Chana "Resource Provisioning in Cloud: State-of-the-Art", Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2012) November 12-14, 2012, University of Victoria, Victoria, Canada.

(Communicated)

Appendix I

Installation of Cloud as IaaS

EUCALYPTUS 3.2

Experimental set up for Cloud as Infrastructure as a Service is been performed by using Eucalyptus.

A.1 Setting up Ubuntu 10.04 LTS for Eucalyptus by installing the Brigde Utils so that the bridge configuration can be done. The following command is used to configure it.

apt-get install bridge-utils

The bridge-utils package will get successfully installed on the Operating System.

A.2 Modify the `/etc/network/interfaces` file as follows

Since the DHCP is used, so the configuration will look as the following

auto lo
iface lo inet loopback
auto br0
iface br0 inet dhcp
bridge_ports eth0

In case of static IP address, the configuration will look similar as follows

auto lo
iface lo
inet loopback
auto br0iface br0 inet static
address <static_IP_address>
network <network>
netmask <netmask>
broadcast <broadcast_IP_address>
gateway <gateway>
bridge_ports eth0
bridge_stp off

After configuring this file reset the network by the command

`/etc/init.d/networking restart`

A.3 Since KVM hypervisor is being used. It can be installed using the following command.

```
aptitude install kvm libvirt-bin ubuntu-vm-builder bridge-utils
```

A.4 It is necessary to get the synchronisation of the various components of the cloud by configuring NTP. This can be done by the help of the command

```
apt-get install openntpd
```

A.5 Now the Operating system is ready for the installation of Eucalyptus. So move the credentials by the following command.

Place the entitlement certificate `/etc/ssl/certs` by following command.

```
mv <license_name>-1.3.0.crt /etc/pki/tls/certs/
```

Place the private key `/etc/ssl/private` by the following command

```
mv <license_name>.key /etc/pki/tls/private/
```

Add the public key to the list of trusted keys:

```
apt-key add c1240596-eucalyptus-release-key.pub
```

A.6 On each server where we want to install Eucalyptus on, go to /etc/apt/apt.conf.d and create a new file (forexample, eucarepo) with the following content

```
Acquire {
https {
VerifyPeer "true";
SslCert "/etc/ssl/certs/<license_name>-1.3.0.crt";
SslKey "/etc/ssl/private/<license_name>.key";
};
};
```

A.7 Create a file in /etc/apt/sources.list.d called eucalyptus-enterprise.list with the following content:

```
deb
https://downloads.eucalyptus.com/software/enterprise/3.0/ubuntu
lucid universe
```

A.8 On all machines that will run either Eucalyptus or Euca2ools, create a file in /etc/apt/sources.list.d called euca2ools.listwith the following content

```
deb
http://downloads.eucalyptus.com/software/euca2ools/2.0/ubuntu
lucid universe
```

A.9 Enter the following command on all machines:

```
apt-get update
```

A.10 Install Eucalyptus packages and dependencies. The following example shows a package install all on the same server.You can install each component on a different server.

```
apt-get install eucalyptus-cloud eucalyptus-cc eucalyptus-sc
eucalyptus-walrus
```

A.11 On each planned NC server, install the NC package:

```
apt-get install eucalyptus-nc
```

A.12 Configuring EUCALYPTUS in MANAGED-NOVLAN mode

In Managed (No-VLAN) mode, Eucalyptus does not use VLANs to isolate the network bridges attached to VMs fromeach other. Configure each CC to use an Ethernet device that lies within the same broadcast domain as all of its NCs.

To configure for Managed (No VLAN) mode:

(i) CLC Configuration:-

No network configuration required.

(ii) CC configuration

Log in to the CC and open the `/etc/eucalyptus/eucalyptus.conf` file.

Go to the Network Configuration section, uncomment and set the following as

```
########################################################################
# GLOBAL CONFIGURATION
########################################################################

# Where Eucalyptus is installed
EUCALYPTUS="/"

# This is the username that you would like eucalyptus to run as
```

```
EUCA_USER="eucalyptus"

# Extra options to pass to the eucalyptus-cloud process, such as log
# levels, heap size, or other JVM flags.
CLOUD_OPTS="--java-home=/usr/lib/jvm/default-java"

#########################################################################
# STORAGE CONTROLLER (SC) CONFIGURATION
#########################################################################

# The number of loop devices to make available at SC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver
# then this setting must be equal to that number.
CREATE_SC_LOOP_DEVICES=256

#########################################################################
# CLUSTER CONTROLLER (CC) / NODE CONTROLLER (NC) SHARED
CONFIGURATION
#########################################################################

# The level of logging output.  Valid settings are, in descending order of
# verbosity, DEBUG, INFO, WARN, ERROR, and FATAL.  The default is DEBUG.
LOGLEVEL="DEBUG"

# The number of old log files to keep when rotating logs.  The default
# is 4.
#LOGROLLNUMBER="4"

# On a NC, this defines the TCP port on which the NC will listen.
# On a CC, this defines the TCP port on which the CC will contact NCs.
NC_PORT="8775"

#########################################################################
# CLUSTER CONTROLLER (CC) CONFIGURATION
#########################################################################

# The TCP port on which the CC will listen.
CC_PORT="8774"

# The scheduling policy that the CC uses to choose the NC on which to
# run each new instance.  Valid settings include GREEDY and ROUNDROBIN.
# The default scheduling policy is ROUNDROBIN.
SCHEDPOLICY="ROUNDROBIN"

# A space-separated list of IP addresses for all the NCs that this CC
# should communicate with.  The ``euca_conf --register-nodes" command
```

```
# manipulates this setting.
NODES="172.31.5.40 172.31.5.98"

# When multiple CCs reside in the same layer 2 broadcast domain, change
# this setting to "Y" to disable tunneling.  This setting has no effect
# in Static or System modes.
#DISABLE_TUNNELING="N"

# The location of the NC service.  The default is
# axis2/services/EucalyptusNC
NC_SERVICE="axis2/services/EucalyptusNC"

# Set this to make the CC cache images, kernels and ramdisks.  NCs must
# be able to reach the CC with the specified value.
#CC_IMAGE_PROXY="ip_of_cc"

# Set this to the location where the CC image proxy should store cached
# images.  The default is /var/lib/eucalyptus/dynserv/
#CC_IMAGE_PROXY_PATH="/var/lib/eucalyptus/dynserv/"

# Set this to the maximum size (in megabytes) of the CC image proxy cache.
# The default is 32768, or 32 gigabytes.
#CC_IMAGE_PROXY_CACHE_SIZE="32768"

################################################################################
# NODE CONTROLLER (NC) CONFIGURATION
################################################################################

# The hypervisor that the NC will interact with in order to manage
# virtual machines.  Supported values include "kvm" and "xen".
HYPERVISOR="kvm"

# The following three options determine whether KVM uses Virtio for
# specific types of I/O with instances.  These options only affect the
# KVM hypervisor.

# If "1", use Virtio for the root file system
USE_VIRTIO_ROOT="0"

# If "1", use Virtio for dynamic block volumes
USE_VIRTIO_DISK="1"

# If "1", use Virtio for the network card
USE_VIRTIO_NET="0"

# The amount of memory, in megabytes, that Eucalyptus is allowed to
```

```
# allocate to instances running on this system.  The default value of
# 0 allows Eucalyptus to use all available memory for instances.
#MAX_MEM="0"

# The number of virtual CPU cores that Eucalyptus is allowed to allocate
# to instances.  The default value of 0 allows Eucalyptus to use all
# CPU cores on the system.
#MAX_CORES="0"

# The amount of disk space, in megabytes, that the NC is allowed to use
# in its work directory ($INSTANCE_PATH/eucalyptus/work).  By default
# the NC chooses automatically.  Values below 10 are ignored.
#NC_WORK_SIZE=50000

# The amount of disk space, in megabytes, that the NC is allowed to use in
# its image cache directory ($INSTANCE_PATH/eucalyptus/cache).  By default
# the NC chooses automatically.  A value below 10 will disable caching.
#NC_CACHE_SIZE=50000

# The number of disk-intensive operations that the NC is allowed to
# perform at once.  A value of 1 serializes all disk-intensive operations.
# The default value is 4.
#CONCURRENT_DISK_OPS=4

# By default, a NC attempts to write the SSH public key associated to
# the instance's filesystem before the instance starts.  A value of 1
# disables this behavior.
#DISABLE_KEY_INJECTION="0"

# The number of loop devices to make available at NC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver then
# this setting must be equal to that number.
#CREATE_NC_LOOP_DEVICES=256

# The directory where the NC will store instances' root filesystems,
# ephemeral storage, and cached copies of images.
INSTANCE_PATH="/var/lib/eucalyptus/instances"

# If euca-bundle-upload, euca-check-bucket, or euca-delete-bundle do
# not appear in the NC's search PATH then specify their locations here.
#NC_BUNDLE_UPLOAD_PATH="/usr/bin/euca-bundle-upload"
#NC_CHECK_BUCKET_PATH="/usr/bin/euca-check-bucket"
#NC_DELETE_BUNDLE_PATH="/usr/bin/euca-delete-bundle"

################################################################################
# NETWORKING CONFIGURATION
```

```
#
# The set of networking settings that apply to a cloud varies based on
# its networking mode.  Each setting in this section lists the modes in
# which it applies.  Unless otherwise noted, all of these settings apply
# only to CCs.  All settings that lack default values must be specified
# in the networking modes that use them.
##############################################################################

# The networking mode in which to run.  The same mode must be specified
# on all CCs and NCs in the entire cloud.  Valid values include SYSTEM,
# STATIC, MANAGED, and MANAGED-NOVLAN.
VNET_MODE="SYSTEM"

# The name of the network interface that is on the same network as
# the NCs.  In Managed and Managed (No VLAN) modes this may need to be
# a bridge.  The default is "eth0".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_PRIVINTERFACE="br0"

# On a CC, this is the name of the network interface that is connected
# to the "public" network.  The default is "eth0".
# Networking modes: Managed, Managed (No VLAN)
#
# On an NC, this is the name of the network interface that is connected
# to the same network as the CC.  Depending on the hypervisor's
# configuration, this may be a bridge or a physical interface that is
# attached to the bridge.  The default is "eth0".
# Networking modes: Managed
VNET_PUBINTERFACE="br0"

# On an NC, this is the name of the bridge interface to which instances'
# network interfaces should attach.  A physical interface that can reach
# the CC must be attached to this bridge.
# Networking modes: System, Static, Managed (No VLAN)
VNET_BRIDGE="br0"

# A map of MAC addresses to IP addresses that Eucalyptus should allocate
# to instances when running in Static mode.  Separate MAC addresses and
# IP addresses with '=' characters.  Separate pairs with spaces.
# Networking modes: Static
#VNET_MACMAP="AA:DD:11:CE:FF:ED=192.168.1.2 AA:DD:11:CE:FF:EE=192.168.1.3"

# A space-separated list of individual and/or hyphenated ranges of public
# IP addresses to assign to instances.
# Networking modes: Managed, Managed (No VLAN)
#VNET_PUBLICIPS="172.31.5.10 172.31.5.11 172.31.5.12 172.31.5.13 172.31.5.14
```

172.31.5.15 172.31.5.16 172.31.5.17 172.31.5.18 172.31.5.19 172.31.5.20 172.31.5.21
172.31.5.22 172.31.5.23 172.31.5.24 172.31.5.25 172.31.5.26 172.31.5.27 172.31.5.28
172.31.5.29 172.31.5.30 172.31.5.31 172.31.5.32 172.31.5.33 172.31.5.34 172.31.5.35
172.31.5.37 172.31.5.38 172.31.5.39 172.31.5.40 172.31.5.41 172.31.5.42 172.31.5.43
172.31.5.44 172.31.5.45 172.31.5.46 172.31.5.47 172.31.5.48 172.31.5.49 172.31.5.50"

The address and network mask of the network the cloud should use for
instances' private IP addresses.
Networking modes: Static, Managed, Managed (No VLAN)
#VNET_SUBNET="10.10.0.0"
#VNET_NETMASK="255.255.0.0"

The number of IP addresses to allocate to each security group.
Specify a power of 2 between 16 and 2048.
Networking modes: Managed, Managed (No VLAN)
#VNET_ADDRSPERNET="32"

The address of the DNS server to supply to instances in DHCP responses.
Networking modes: Static, Managed, Managed (No VLAN)
#VNET_DNS="172.31.1.6"

The network broadcast address and default gateway to supply to instances
in DHCP responses.
Networking modes: Static
#VNET_BROADCAST="172.31.1.1"
#VNET_ROUTER="192.168.1.1"

Set this to the IP address that other CCs can use to reach this CC
if layer 2 tunneling between CCs does not work. It is not normally
necessary to change this setting.
Networking modes: Managed, Managed (No VLAN)
#VNET_LOCALIP="your-public-interface's-ip"

# The ISC DHCP server executable to use. The default is
# "/usr/sbin/dhcpd3".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_DHCPDAEMON="/usr/sbin/dhcpd3"

# The user as which the DHCP daemon runs on your distribution.
# The default is "dhcpd".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_DHCPUSER="dhcpd"

(iii) NC Configureation `/etc/eucalyptus/eucalyptus.conf` file.

Go to the Network Configuration section, uncomment and set the following as

```
##########################################################################
# GLOBAL CONFIGURATION
##########################################################################

# Where Eucalyptus is installed
EUCALYPTUS="/"

# This is the username that you would like eucalyptus to run as
EUCA_USER="eucalyptus"

# Extra options to pass to the eucalyptus-cloud process, such as log
# levels, heap size, or other JVM flags.
CLOUD_OPTS=""

##########################################################################
# STORAGE CONTROLLER (SC) CONFIGURATION
##########################################################################

# The number of loop devices to make available at SC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver
# then this setting must be equal to that number.
#CREATE_SC_LOOP_DEVICES=256

##########################################################################
# CLUSTER CONTROLLER (CC) / NODE CONTROLLER (NC) SHARED
CONFIGURATION
##########################################################################

# The level of logging output.  Valid settings are, in descending order of
# verbosity, DEBUG, INFO, WARN, ERROR, and FATAL.  The default is DEBUG.
LOGLEVEL="DEBUG"

# The number of old log files to keep when rotating logs.  The default
# is 4.
#LOGROLLNUMBER="4"

# On a NC, this defines the TCP port on which the NC will listen.
# On a CC, this defines the TCP port on which the CC will contact NCs.
NC_PORT="8775"

##########################################################################
# CLUSTER CONTROLLER (CC) CONFIGURATION
##########################################################################

# The TCP port on which the CC will listen.
CC_PORT="8774"
```

```
# The scheduling policy that the CC uses to choose the NC on which to
# run each new instance.  Valid settings include GREEDY and ROUNDROBIN.
# The default scheduling policy is ROUNDROBIN.
SCHEDPOLICY="ROUNDROBIN"

# A space-separated list of IP addresses for all the NCs that this CC
# should communicate with.  The ``euca_conf --register-nodes" command
# manipulates this setting.
NODES=""

# When multiple CCs reside in the same layer 2 broadcast domain, change
# this setting to "Y" to disable tunneling.  This setting has no effect
# in Static or System modes.
#DISABLE_TUNNELING="N"

# The location of the NC service.  The default is
# axis2/services/EucalyptusNC
NC_SERVICE="axis2/services/EucalyptusNC"

# Set this to make the CC cache images, kernels and ramdisks.  NCs must
# be able to reach the CC with the specified value.
#CC_IMAGE_PROXY="ip_of_cc"

# Set this to the location where the CC image proxy should store cached
# images.  The default is /var/lib/eucalyptus/dynserv/
#CC_IMAGE_PROXY_PATH="/var/lib/eucalyptus/dynserv/"

# Set this to the maximum size (in megabytes) of the CC image proxy cache.
# The default is 32768, or 32 gigabytes.
#CC_IMAGE_PROXY_CACHE_SIZE="32768"

################################################################################
# NODE CONTROLLER (NC) CONFIGURATION
################################################################################

# The hypervisor that the NC will interact with in order to manage
# virtual machines.  Supported values include "kvm" and "xen".
HYPERVISOR="kvm"

# The following three options determine whether KVM uses Virtio for
# specific types of I/O with instances.  These options only affect the
# KVM hypervisor.

# If "1", use Virtio for the root file system
USE_VIRTIO_ROOT="0"
```

```
# If "1", use Virtio for dynamic block volumes
USE_VIRTIO_DISK="1"

# If "1", use Virtio for the network card
USE_VIRTIO_NET="0"

# The amount of memory, in megabytes, that Eucalyptus is allowed to
# allocate to instances running on this system.  The default value of
# 0 allows Eucalyptus to use all available memory for instances.
#MAX_MEM="0"

# The number of virtual CPU cores that Eucalyptus is allowed to allocate
# to instances.  The default value of 0 allows Eucalyptus to use all
# CPU cores on the system.
#MAX_CORES="0"

# The amount of disk space, in megabytes, that the NC is allowed to use
# in its work directory ($INSTANCE_PATH/eucalyptus/work).  By default
# the NC chooses automatically.  Values below 10 are ignored.
#NC_WORK_SIZE=50000

# The amount of disk space, in megabytes, that the NC is allowed to use in
# its image cache directory ($INSTANCE_PATH/eucalyptus/cache).  By default
# the NC chooses automatically.  A value below 10 will disable caching.
#NC_CACHE_SIZE=50000

# The number of disk-intensive operations that the NC is allowed to
# perform at once.  A value of 1 serializes all disk-intensive operations.
# The default value is 4.
#CONCURRENT_DISK_OPS=4

# By default, a NC attempts to write the SSH public key associated to
# the instance's filesystem before the instance starts.  A value of 1
# disables this behavior.
#DISABLE_KEY_INJECTION="0"

# The number of loop devices to make available at NC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver then
# this setting must be equal to that number.
CREATE_NC_LOOP_DEVICES=256

# The directory where the NC will store instances' root filesystems,
# ephemeral storage, and cached copies of images.
INSTANCE_PATH="/var/lib/eucalyptus/instances"
```

```
# If euca-bundle-upload, euca-check-bucket, or euca-delete-bundle do
# not appear in the NC's search PATH then specify their locations here.
#NC_BUNDLE_UPLOAD_PATH="/usr/bin/euca-bundle-upload"
#NC_CHECK_BUCKET_PATH="/usr/bin/euca-check-bucket"
#NC_DELETE_BUNDLE_PATH="/usr/bin/euca-delete-bundle"

##############################################################################
# NETWORKING CONFIGURATION
#
# The set of networking settings that apply to a cloud varies based on
# its networking mode.  Each setting in this section lists the modes in
# which it applies.  Unless otherwise noted, all of these settings apply
# only to CCs.  All settings that lack default values must be specified
# in the networking modes that use them.
##############################################################################

# The networking mode in which to run.  The same mode must be specified
# on all CCs and NCs in the entire cloud.  Valid values include SYSTEM,
# STATIC, MANAGED, and MANAGED-NOVLAN.
VNET_MODE="SYSTEM"

# The name of the network interface that is on the same network as
# the NCs.  In Managed and Managed (No VLAN) modes this may need to be
# a bridge.  The default is "eth0".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_PRIVINTERFACE="eth0"

# On a CC, this is the name of the network interface that is connected
# to the "public" network.  The default is "eth0".
# Networking modes: Managed, Managed (No VLAN)
#
# On an NC, this is the name of the network interface that is connected
# to the same network as the CC.  Depending on the hypervisor's
# configuration, this may be a bridge or a physical interface that is
# attached to the bridge.  The default is "eth0".
# Networking modes: Managed
VNET_PUBINTERFACE="eth0"

# On an NC, this is the name of the bridge interface to which instances'
# network interfaces should attach.  A physical interface that can reach
# the CC must be attached to this bridge.
# Networking modes: System, Static, Managed (No VLAN)
VNET_BRIDGE="br0"

# A map of MAC addresses to IP addresses that Eucalyptus should allocate
# to instances when running in Static mode.  Separate MAC addresses and
```

```
# IP addresses with '=' characters.  Separate pairs with spaces.
# Networking modes: Static
#VNET_MACMAP="AA:DD:11:CE:FF:ED=192.168.1.2 AA:DD:11:CE:FF:EE=192.168.1.3"

# A space-separated list of individual and/or hyphenated ranges of public
# IP addresses to assign to instances.
# Networking modes: Managed, Managed (No VLAN)
#VNET_PUBLICIPS="your-free-public-ip-1 your-free-public-ip-2 ..."

# The address and network mask of the network the cloud should use for
# instances' private IP addresses.
# Networking modes: Static, Managed, Managed (No VLAN)
#VNET_SUBNET="192.168.0.0"
#VNET_NETMASK="255.255.0.0"

# The number of IP addresses to allocate to each security group.
# Specify a power of 2 between 16 and 2048.
# Networking modes: Managed, Managed (No VLAN)
#VNET_ADDRSPERNET="32"

# The address of the DNS server to supply to instances in DHCP responses.
# Networking modes: Static, Managed, Managed (No VLAN)
#VNET_DNS="your-dns-server-ip"

# The network broadcast address and default gateway to supply to instances
# in DHCP responses.
# Networking modes: Static
#VNET_BROADCAST="192.168.1.255"
#VNET_ROUTER="192.168.1.1"

# Set this to the IP address that other CCs can use to reach this CC
# if layer 2 tunneling between CCs does not work.  It is not normally
# necessary to change this setting.
# Networking modes: Managed, Managed (No VLAN)
#VNET_LOCALIP="your-public-interface's-ip"

# The ISC DHCP server executable to use.  The default is
# "/usr/sbin/dhcpd3".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_DHCPDAEMON="/usr/sbin/dhcpd"

# The user as which the DHCP daemon runs on your distribution.
# The default is "dhcpd".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_DHCPUSER="dhcpd"
```

A.13 Star CLC and Log in to the CLC.

Enter the following command to initialize the CLC:

```
/usr/sbin/euca_conf –initialize
```

It will take few minutes.

Start the various components by

```
service eucalyptus-cloud start
service eucalyptus-cc start
```

A.14 To register Walrus:

On the CLC server, enter the following command:

```
/usr/sbin/euca_conf --register-walrus --partition walrus --host
<walrus_IP_address> --component <walrus_name>
```

A.15 To register the CC:

On the CLC, enter the following command:

```
/usr/sbin/euca_conf --register-cluster --partition
<partition_name> --host <CC_IP_address> --component <cc_name>
```

A.16 To register the SC:

On the CLC, enter the following command:

```
/usr/sbin/euca_conf --register-sc --partition <partition_name> -
-host <SC_IP_address> --component <SC_name>
```

A.17 On a CC, register all NCs using the following command with the IP address of each NC server:

```
/usr/sbin/euca_conf --register-nodes "<node0_IP_address> ...
<nodeN_IP_address>"
```

A.18 Generating Administration Credentials

Generate administrator credentials.

```
/usr/sbin/euca_conf --get-credentials admin.zipunzip admin.zip
```

Source the eucarc file.

```
source eucarc
```

Now EUCALYPTUS Cloud is ready to use.