Edwin Guchu

Implementation of cloud infrastructure using open source software

Helsinki Metropolia University of Applied Sciences
Bachelor of Engineering
Degree Programme in Information Technology
Thesis
23rd March 2012

Author Title Number of Pages Date	Edwin Guchu Implementation of cloud infrastructure using open source software 43 pages + 7 appendices 23rd March 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Instructor(s)	Sakari Lukkarinen, Lecturer

The purpose of this thesis is to demonstrate and show the technical and logical part of building an open source cloud. The process of doing this took two main different approaches in making it work. This thesis will explain and demonstrate the two ideas which were diskless booting distributing computing and Ubuntu Enterprise Cloud based on Eucalyptus. In order to understand the technologies used, a project on building the cloud was done at the premises of the Metropolia University of Applied Sciences in Leppävaara, Espoo.

In this project, the installation and configuration were done on two Ubuntu servers and one client Ubuntu desktop computer in both approaches. As the project went on, challenges were encountered which caused the change to different ways of facing the aim and the objective of the project. Regarding the first approach of building a cloud computing environment, explanations on how the idea came about, how to use preboot execution environment, booting through the network and managing the cluster computers remotely from the servers are all included in this thesis. Drawbacks of this approach are also explained in this documentation.

Regarding the second approach, Ubuntu Enterprise Cloud based on Eucalyptus was used as the preferred technology. This thesis will explain, demonstrate and show the process and the outcome of using this technology. Virtualization Technology and distributed computing among other techniques were used in this approach. In addition to this, a simple demonstration of the user interface will be discussed.

The results of this thesis show the best and most efficient method of building a cloud, private, public or hybrid. The results also show how to run an open source based cloud computing environment and how to use it as a new beginner in the cloud environment. Open source based cloud computing infrastructures are currently implemented in many different ways. This thesis illustrates one of the implementations.

Keywords	Cloud computing, open source, Eucalyptus, Virtualisation Technology, PXE

Contents

Abbreviations and Terms

Cloud computing environment

Is Internet computing whereby servers provide resources, software and data to computers and other services on demand.

Amazon Web Service (AWS)

Is a collection of web services that together make up a cloud computing platform.

Preboot eXecutional Environment (PXE)

Is the basic interface that allows computers with no operating systems to load, to be configured and booted remotely by an administrator from the network.

Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems).

Is an open source software platform that implements cloud services.

DHCP (Dynamic Host Configuration Protocol)

Is a network configuration protocol that hosts the Internet Protocol.

TFTP (Trivial File Transfer Protocol)

Is a file transfer protocol used by boot files between machines in a local network.

NFS (Network File System, protocol)

Is a sharing file system protocol that allows a user on a client computer to access files from an allowed server.

BIOS (Basic Input/ Output Operating System)

Is the first code that is run when the PC is powered on.

UEC (Ubuntu Enterprise Cloud)

Is a cloud computing module that provides distributions of a Linux based operating system.

EC2 (Elastic Computing Cloud)

Is a type of cloud services that delivers scalable computing capacity in a cloud.

S3 (Amazon Simple Storage Service)

Is an online storage infrastructure used to store and retrieve data at any given time anywhere connected to the Internet.

KVM (Kernel Virtual Machine)

Is virtualization software used by Linux kernel.

API (Application Programming Interface)

Is a source code in programming designed to be used as an interface by software components to interact with each other.

NC (Node Controller)

Is a server used to host virtual machines in Eucalyptus and Ubuntu Enterprise Cloud.

CC (Cluster Controller)

Is a control computer that manages several peripheral node controllers.

CLC (Cloud Controller)

Is the entry-point into the cloud for administrators, project managers and cloud users. It is responsible for querying the nodes for resources.

WS3 (Walrus Simple Storage Services)

Is a storage service in Eucalyptus that is compatible with Amazon's S3. It allows users to store data in bucket–based storage services.

EMI (Eucalyptus Machine Image)

Is a combination of a Virtual disk image, kernel and ramdisk images as a virtual machine.

1 Introduction

In the last five years, IT companies have been working hard and investing much in creating and developing a cloud computing environment. The cause of this is high demand and greater competition in the innovative world of computer industry. Due to high demand of software and applications, companies ought to have their software and projects online, thus the need of building an efficient cloud computing environment. In the current days of the Internet, high network speeds, efficiency and security have been perfected to greater heights. Thus, almost everyone can now upload sensitive details over the network.

As companies continue to evolve with the latest technology and businesses are approaching the international market, the individuals in different sectors in the international companies would like to work on their projects or software from wherever they are apart from their offices. Here is where cloud computing comes in. It would enable companies and individuals to use the same software and platforms from wherever they are connected to the Internet and not tunneling to their office computers. Currently, many databases, applications and software are found on the Internet and they are owned by different companies. Web developers and software engineers can use development tools on the Internet without downloading, and users can use applications on the Internet from anywhere in the world. This is a wave that has and will continue fascinating many.

The purpose of this project has been to show how to build a cloud computing environment using open source software and what to consider while determining the perfect software to use. The project was initiated by the Metropolia University of Applied Sciences and the aim was to build a fully functional computer cloud system. Interest in the project arose due to the fact that cloud computing is part of the latest technology. Another reason was to gain the understanding of how cloud computing works and to learn how to build and maintain a functional system. There was a chance to build a cloud from scratch using open source software without any complicated hardware and also to add more experience on Linux distribution.

The scope of this project was limited to a description of setting up the environment required for this project in the University of Applied Sciences. In this case, the project was mostly focused on the infrastructure. Also, a requirement was the testing of all the functionality of this cloud by using a physics and mathematical simulation software called COMSOL.

However, a few limitations were encountered. In this project, troubleshooting and research in the two approaches took most of the project time. The reason for this was that the project members had less experience on these approaches though they had the ideas of how to work towards achieving the goal of the project. Due to these limitations, the testing part of COMSOL software on these two approaches was left out of this thesis.

2 Basic cloud computing infrastructure

2.1 Cloud computing overview

Cloud computing has many definitions but in this thesis cloud computing is understood to be the delivery of computing services rather than computing products. It can also be explained as internet computing whereby shared servers provide resources, software and data to computers and other devices on demand through the network (i.e. the Internet) [1]. Apart from the World Wide Web (www) structure, the cloud structure is made up of hardware, networks, storage, services and interfaces that enable the delivery of computing as a service. Cloud services include the delivery of software, infrastructure and storage over the Internet based on user demand.

Distributed computing refers to the use of different resources from different computers being used simultaneously to solve computational problems. In distributed computing, problems are divided into many tasks, each of which is solved by one or more computers. The understanding of a cloud computing environment brought with it many ideas and approaches of using computers working together simultaneously to perform a common task. As cloud computing becomes a more integral part of the IT world, many companies have invested much in building up public and private clouds. Companies such as Amazon, Google, Microsoft, Salesforce, Skytap and Rackspace Cloud have laid their ground on providing cloud computing services to companies and individuals. [2.]

Most of these cloud computing environments are built on Virtualization (Hypervisor) technology, for example Amazon Web Service (AWS). Virtualization provides a means to separate a computer's physical hardware, the operating system and applications by simulating software. This software hypervisor is installed into the computer. The software also uploads files that define a virtual computer. A virtual appliance is an application that is grouped together with all the components that it needs in order to run with an operating system [3].

The virtualization of computers and operating systems hides the physical characteristics of computers to the users. The hypervisor is part of virtualization, which allows many virtual operating systems to run on the same computer simultaneously. With this software in mind, companies and research institutes widen the cloud computing field for more virtual machines to work together to build a bigger cloud with better functionalities.

The world of cloud computing has different parties involved [4]:

- The end user who does not really have to know anything about the underlying technology.
- The business management who needs to take responsibility for overall governance of data or services living in a cloud.
- The cloud service provider who is responsible for IT hardware and maintenance.

Figure 1 below illustrates the generation phases of computing since the beginning of its development to the present technology. These different technologies truly explain where the idea of cloud computing came from and if there is room for more phases.

In phase 1, powerfull mainframes were used by many users through dummy terminals. In phase 2, stand-alone PCs became powerful enough to meet the needs of the users and so the development of the PC computers continued up to date. In phase 3, servers, laptops, PCs and printers were all connected together through a local network to share resources and increase performance and conveniences. In phase 4, local networks were connected together with other networks to form a global network called the Internet. In phase 5, grid computing provided shared computing power and storage through a distributed computing system. The current development is at phase 6. This phase looks similar to the grid phase but the cloud phase is bigger with more servers, more capacity. It is also open to the world and mostly secure. In the cloud phase, most resources are shared on the Internet in a scalable and simple way [2].

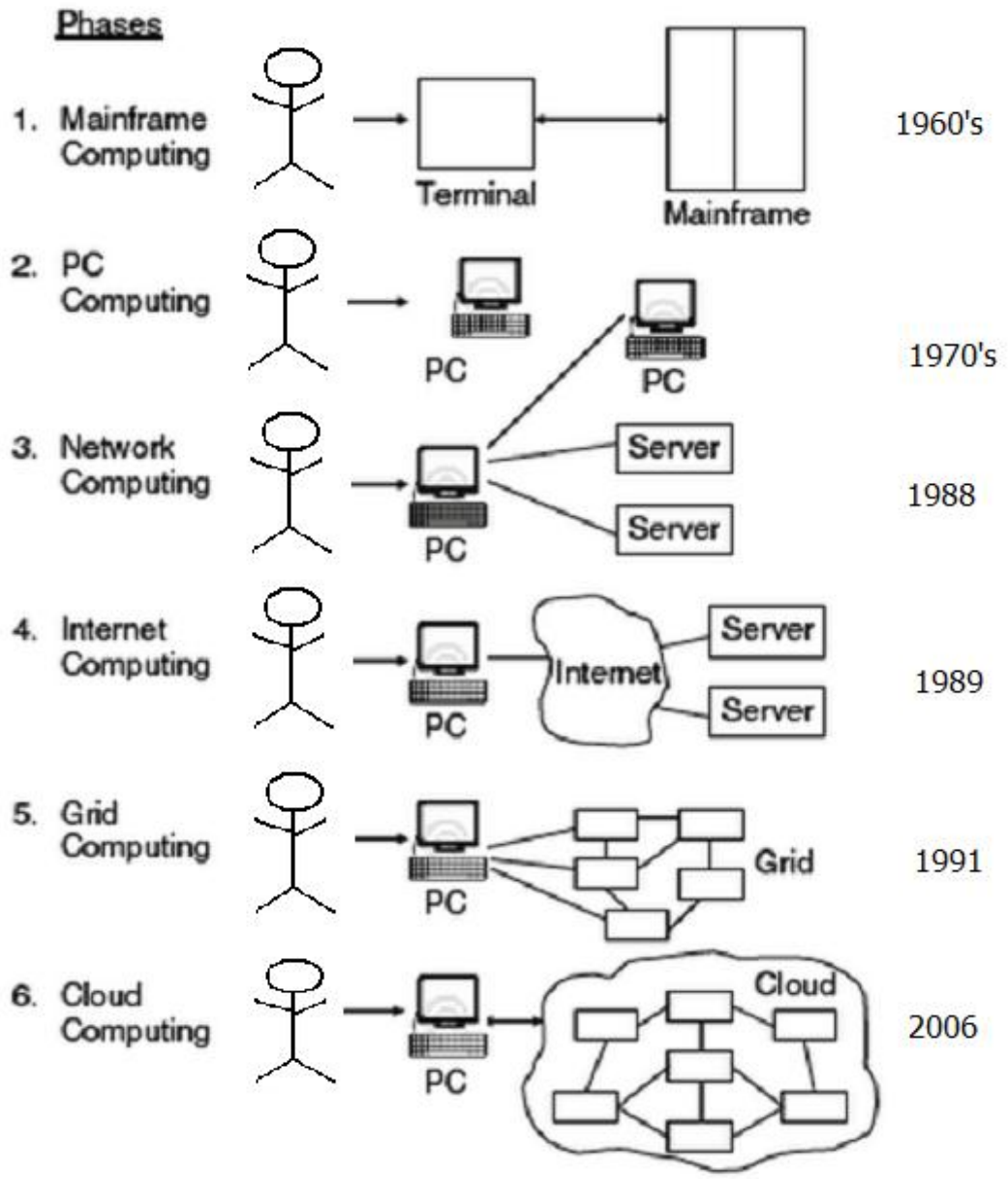

Figure 1. Six phases of computer paradigm (Adapted from[4,4]).

There are basic features that need considerations when building an efficient cloud for an institute or a company. These are listed below [1]:

- **Elasticity and scalability.** This means that the service needs to be available at all times and to be designed to scale upward for high periods of demand and downward for lighter ones. The cloud should be able to scale when additional users are added and when the application requirements change.

- **Application programming interfaces (APIs).** APIs need to be standardized for cloud services. These interfaces provide the instructions on how two applications or data sources can communicate with each other. A standardized interface lets the customer link a cloud service with ease instead of resorting to custom programming.
- **Measurement and performance monitoring.** This is a feature with service management that monitors and measures the working condition, thus maintaining the required service level for the organization.
- **Security.** Is an essential factor in a cloud. Handing over critical information or application infrastructure to a cloud service provider requires making sure that the information cannot be compromised.
- **Billing and metering.** These services should be a built-in service that bills customers. This is common for the public clouds.
- **Self-service provisioning.** This should enable customers to easily get cloud services without going through a lengthy process.

2.1.1 Service delivery models in computer clouds

Figure 2 shows the three distinct models of cloud services: Infrastructure as a Service, Platform as a Service, and Software as a Service. In reality, the lines between the different delivery models are often thin. For example, a Software as a Service (SaaS) vendor might decide to offer separate infrastructure services to customers.

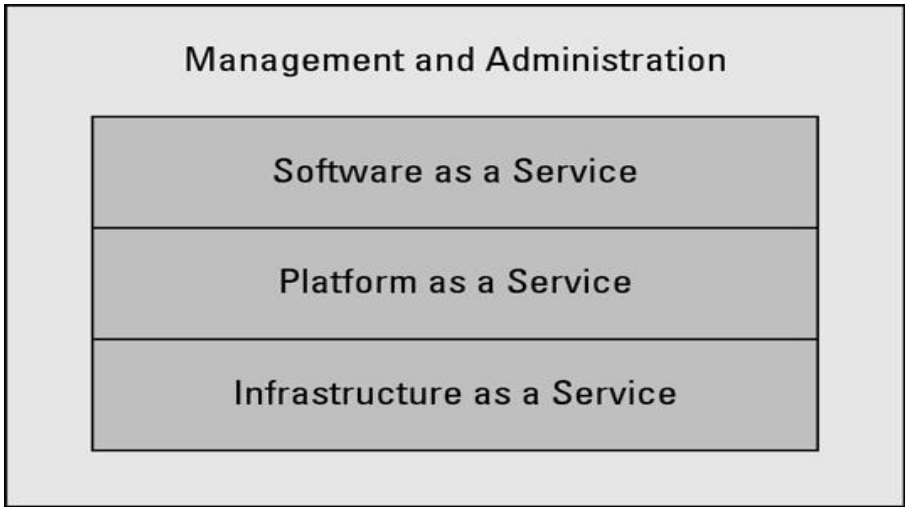

Figure 2.Types of cloud services (Adapted from [5]).

Software as a Service (SaaS) is a cloud delivery model that has existed for the last eleven years. A SaaS is an implementation of an application or process that is developed on a cloud platform and hosted in a cloud infrastructure. An example of this kind of a service is the Microsoft cloud. A Microsoft cloud user can order a Microsoft office space and use Microsoft Word for his/her documentation through the Internet and then save it under the cloud storage services. A SaaS provider on the other hand, delivers domain-specific applications or services over the Internet and charges the end users on a pay-per-usage basis. There are two types of SaaS providers on the Internet as explained below.

- **Simple multi-tenancy:** Each customer has its own resources that are segregated from those of other customers. This amounts to a relatively inefficient form of multi-tenancy.
- **Fine-grain multi-tenancy:** This offers the same level of segregation but is far more efficient. All resources are shared, but customer data and access capabilities are segregated within the application. [5.]

Infrastructure as a Service (IaaS) is the delivery of computer hardware (servers, networking technology, storage, and data center space) as a service. This may also include the delivery of operating systems and virtualization technology to manage the resources. Instead of going through capacity planning, installation and configuration processes, the IaaS model allows a cloud user or customer to start a new project quickly by renting computing resources. The key characteristic of an IaaS cloud is elasticity and scalability, enabling computing resources to scale up and down. Most IaaS cloud providers offer scalability under customer control with direct self-service interfaces, through which consumers can request to scale, control, and manage computing resources. An IaaS cloud is also referred as a resource cloud. According to the different types of resources offered, IaaS cloud can be further divided into three sub-categories: Computing as a Service (CaaS), Storage as a Service, and Database as a Service (DaaS).

Platform as a Service (PaaS) includes the delivery of more than just infrastructure. It delivers a solution stack or an integrated set of software that provides everything a developer or software engineer needs to build an application (i.e. design, development, debugging, testing, and deployment). Most PaaS vendors lock developers into particular development platforms and debugging tools. However, they do not allow direct communication with lower computing infrastructures. Some certain programming applications might be provided with limited functionalities of infrastructure control and management.

All the cloud services models function differently from each other though all of them working hand in hand with each other. Table 1 shows the different technologies in cloud computing service models. With different cloud services providers, they have different technologies they use in supporting their service models. Table 1 illustrates the kind of applications used in different service model.

Table 1. Types of service delivery models (Adapted from [4,25]).

Service Type	**IaaS**	**PaaS**	**SaaS**
Service category	VM rental, online storage	Online operation, online database, messaging, queue	Application and software rental
Service customization	Server template	Logic resource template	Application template
Service accessing and using	Remote console, web 2.0	Development tools and cloud	Web 2.0
Service monitoring	Physical resources monitoring	Logic resource monitoring	Application monitoring
Service measurements	Physical resource metering	Logic resource usage metering	Business resource usage metering
Service security	Storage encryption and isolation, SSL/SSH	Data isolation operation environment isolation, SSL	Data isolation operation environment isolation, SSL, web authentication and authorization

Service categories are divided into three models. In Iaas, virtual machines and storage capacities are rented out to a cloud user and the user can decide on what to do with it. On the other hand, Paas and SaaS are more on software and applications rentals. Service access and usage are also different in these models. In Iaas, remote access via the console is applicable and also through the Web 2.0.

2.2 Approaches of Architecture

The IT department of the Metropolia University of Applied Sciences delivered 100 standard desktop computers for this project. These computers are the normal Desktop PCs which were previously used in the classrooms, and the idea was to build a fully functional cloud computing structure using these computers. Apart from the computers, there were also two network switches supplied for the project.

Some of the factors which were put into consideration were how many resources there were in these computers (i.e. memory, processor power, hard disk space) and what the power consumption of all these computers was. Other considerations were how much work force was needed to monitor the machines, even in terms of maintenance and troubleshooting and the kind of software to be used in the building of the cloud. According to the criteria of this project, the cloud is required to be open source based. Open source software is computer software that is available in source code and has a right reserved for copyright holders. It is provided under a free software license that permits users to read, study, improve and also distribute the software [6]. Open source software is mostly not commercial, meaning it can be downloaded for free from the Internet. Such software is often developed and improved by the public. In this project, no commercial software was used apart from COMSOL, whereby Metropolia got a license to use the software for this project.

Considering the criteria used in this project, there were a variety of approaches in building this cloud. Here are demonstrations of the two ideas used in building this cloud:

- Preboot Executional Environment (PXE).
- Ubuntu Enterprise Cloud (UEC) and based on Eucalyptus.

The above techniques are demonstrated in the next chapters. The preferred working technique helped the project to fulfil its aim.

2.2.1 PXE Diskless booting

The Preboot Execution Environment (PXE) is an industry standard client/server interface that allows networked computers that are not yet loaded with an operating system to be configured and booted remotely by an administrator.

The PXE code is typically delivered to a boot disk that allows a computer to communicate with the network server so that the machine can be remotely configured and its operating system can be remotely booted. Most computers sold since the year 2001 are implemented with two useful features on the Ethernet interfaces: wake–on–LAN and network boot [7].

Wake–on–LAN: Wake on Local Area Network is the ability to switch on or off a remote computer through special network packets from the administrative server within the local network. This only works with network cards and motherboards that are wake–on–LAN compliant. For example, when the PC shuts down, the Network Interface Card (NIC) still gets power, and keeps listening on the network for the special packet to arrive.
This packet must contain a certain byte-sequence but can be encapsulated in any kind of packet, for example IP packets.

Network boot: There are several ways computers can boot over a network, but the main and most used is PXE. PXE is a Dynamic Host Configuration Protocol (DHCP) extension and all that is needed is an up-to-date DHCP server and a Trivia File Transfer Protocol (TFTP) server. A DHCP server and a TFTP server can be set up to handle PXE boot requests. [7.]

When a PXE boot is in action, there are three major steps provided:

1) The Dynamic Host Configuration Protocol (DHCP), which allows a cluster computer to receive an IP address to gain access to the network servers.

2) A set of application program interfaces (API) that are used by the cluster's Basic Input/Output Operating System (BIOS) or a downloaded Network Bootstrap Program (NBP) that automates the booting of the operating system and other configuration steps.

3) A standard method of initializing the PXE code in the PXE ROM chip or boot disk.

The advantages of using PXE include:

- The cluster machine does not necessarily need an operating system or even a hard disk.
- The machine can be rebooted in the event of hardware or software failure. This allows the administrator to diagnose and perhaps fix the problem.
- Since PXE is vendor-independent, new types of computers can easily be added to the network [8].

Currently, some IT companies use PXE with their computer systems.

2.2.2 UEC and Eucalyptus

Eucalyptus is a software platform in Ubuntu for the implementation of private and public cloud computing on cluster computers. It is open source software which began as a research project in the University of California, the USA. It exports a user interface that is compatible with the Amazon EC2 (elastic cloud computing) compatible and S3 (Storage 3) cloud platform. Eucalyptus stands for (Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems). It uses a variety of virtualization technologies including VMware, Xen and KVM hypervisors to implement the cloud abstractions it supports. In this project the KVM hypervisor (Kernel based virtual machine) was used in building the virtual machine which will be used by the Eucalyptus.

The Eucalyptus cloud computing platform has five high-level components:

- The Walrus storage controller which provides a basic storage mechanism for persistent storage and access control of virtual machine images and user data.
- The Storage controller (SC) that provides the storage uses by the virtual machines.
- The Cloud controller (CLC) provides a compliant web interface to manage the Eucalyptus infrastructure.
- The Node controller (NC) controls the Virtual Machines, inspects and terminates the VM instances.
- A Cluster controller (CC) controls the whole virtual network, the relation between the running nodes and also the relation between the instances and the external user.

All these components can work simultaneously either from a single server or multiple servers.

3 PXE Diskless Booting

3.1 Overview and Design

The idea here is to have this kind of a cloud where all the 100 computers that will be in our cloud are under one master server that will be able to manage this cloud. All the computers working together form a cluster. This cluster of computers have only the hardware parts (CPU boxes) connected to a switch via the Ethernet cable and plugged to the power source, in this case an electric switch. They will be booting from the network and everything else is done from the administrative (master) server. The point is to utilize the resources from this cluster in a distributional way, in this case the Central Processing Unit (CPU) capacity, hard disk space, and the memory capacities of these computers. The technology used is distributed computing. The cluster computers will be booting from a master server which will be having the Debian (Ubuntu 10.04 LTE) Image for the cluster to boot from. The process of a computer booting from the network is diskless booting. For a complete diskless boot, the cluster computers need to have a network card that is compatible with PXE (Preboot execution Environment).

The server will have to run a Trivial File Transfer Protocol (TFTP) daemon to share the kernel/boot sector along with NFS (Network File System, protocol) to share the root directory. The configuration of the Trivial File Transfer Protocol (TFTP), NFS (Network File System, protocol) and Dynamic Host Configuration Protocol (DHCP) are configured to one server which is the master server. The COMSOL software will then be installed into the cluster computers through the network from the server that is running the Debian based Ubuntu Server 10.04 LTE Operating System. COMSOL Multiphysics software is a finite element analysis, solver and simulation software for various physics, engineering applications and complex mathematics. COMSOL can run a task on many cores in parallel (shared -memory processing) [9].

All 100 computers are connected to the university internal private IP network through the network switches to the server. A functioning cloud should perform this kind of a task. When a complex mathematical or physics problem is submitted to the COMSOL simulator on the server, the server can distribute the problem to the cloud (cluster computers) and the computers will all do different tasks in solving the different parts of the problem.

Then they will send back the solutions to the server and the server will compile the solutions it gets from the cluster to form one conclusive answer to the complex query. This is the idea of the distribution computing in the cloud. The forwarding and passing of these problems through the network is done by the NFS network protocol.

3.2 Topology

According to the topology and the structure of the laboratory room in which the project was being carried out at Metropolia, there were some limitations with the size of the room. Because of this reason, the best and more reasonable way to proceed on with the project was to use only three computers from the 100 computers. From the three machines, one represents the master server where the Ubuntu server 10.10 LTE was installed and the other two worked as cluster computers which perform PXE booting from the server. Figure 3 shows the topology design for the PXE booting system.

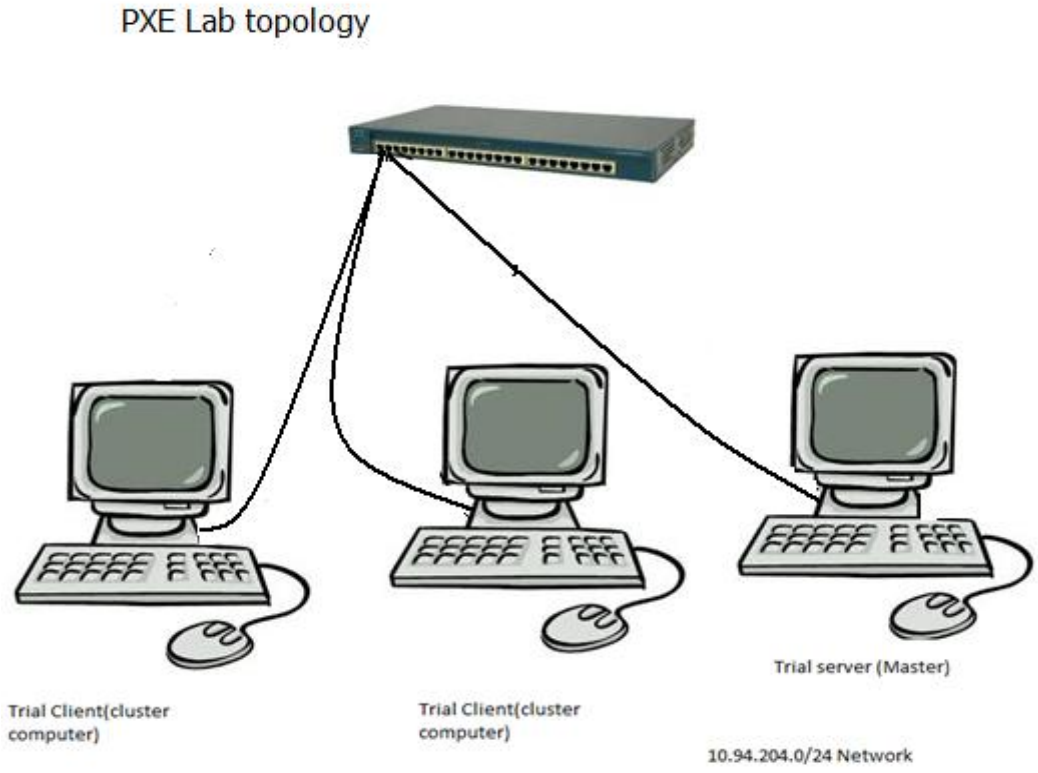

Figure 3. The topology setup in the lab room at Metropolia Leppaävaara campus

According to figure 3 above, the idea of this topology is to configure the master server to be a PXE (Preboot eXecution Environment) boot server. In the project, the server performed the Wake-on-LAN and network booting onto the cluster computers.

In general, for the cluster computers to perform the network booting, the BIOS for these computers needs to be enabled to Wake-on-LAN. To add on this, the cluster boot sequence should also be adjusted to PXE options as the first boot sequence. Table 2 summarizes the benefits of PXE diskless booting.

Table 2 Benefits of PXE Diskless booting

Benefits of PXE diskless booting topology in the project:
-Saves power consumption.
-Reduces time to configure individual PCs.
-Easier to monitor the 100 computers from the server.
-Easier to maintain and troubleshoot the private cloud.

PXE is the most efficient and convenient way for dealing with a large scale of cluster computers.

3.3 Installation processes on the server

Table 3 describes a list of the minimum requirements needed on a server in the installation of the server applications:

Table 3 List of minimum requirements on a server for the installation of PXE boot

Requirement
• An operating system that can support all the server applications, for example Ubuntu 11.04 • An Ubuntu system with nfs-kernel-server, dhcp-server, tftpd server (the server) • PXE-bootable system (the clients) • A minimum disk space of 50 GB on the server to hold the client file system • A minimum of 2mbps network connection between the client and the server

The features of the master server were the following: Hard disk 250 Gb, RAM memory 2Gb, Dual Core processor 2.26 GHz, Two Network Interface Card (eth0 and wireless NIC Wlan). According to the topology in figure 3, the server is running Ubuntu server 10.10 LTE and the installation of this operating system was done from a bootable CD. During the installation of this OS in the server, the network cards were enabled so as for internet access. The wireless network card was for accessing the Internet and the other interface (eth0) is for the cloud network.

Through this interface, all the requirements of the cloud will be passed through this interface. The IP address allocated for it was a private network 10.94.204.0/24 and on the eth0 interface was configured as 10.94.204.1/24. In this private network address, there were 254 IP addresses for this project. In this PXE technique, the first few addresses are used.

When the OS installation is complete, and the server is connected to the Internet via Wlan, the next step is to update the operating system. This process is important because of compatibility issues. Since every system has different types of hardware, the updating of the OS is always the first step after the OS is installed and running. After the updating process, the OS will run efficiently according to the system it has been installed in. The command used to update is *sudo apt-get update.*

While all the basic installation and the updating are done, the following installations are performed to achieve diskless booting over the network:

- DHCP Server
- TFTP server: tftp1
- NFS server: nfs1
- PXE image location on NFS server

The first step in PXE booting is the installation and the configuration on the DHCP server. The command to install is *sudo apt-get install dhcp3-server*. During the installation, a dhcp file system is installed automatically. After the installation, configuration to the dhcp file system is set up to offer a /tftpboot/pxelinux.0 as a boot file. To add on this configuration, a set of IP addresses are set for the client computers, in this case, addresses from 10.94.204.100 to 10.94.204.200 are set for the dhcp clients. However, for this set up, it is easier to assign fixed IP addresses to the PXE client machines. Also to add on this configuration, it is advisable to include the MAC address and the IP address of the server as hardware Ethernet identity. The configured file is *nano /etc/dhcp3/dhcpd.conf* and the configured set up can be found in appendix 1. After the entire configuration on dhcp is done, restarting dhcp is the next step and it is done by */etc/init.d/dhcp3-server restart.*

The next configuration is the TFTP server set up. The purpose of the TFTP server in PXE approach is to allow and make room for a root directory to be installed and run on it. To get this protocol server on the project server, the *sudo apt-get install tftp-hpa* command is used to download and install it. Modification of the root directory is necessary so as to make this file run as a root directory.

This is done by *nano/etc/default/tftpd-hpa* file. This file name is *tftp-hpa* file and the change is to edit the tftp directory part into "/srv/tftp". After the changes, it is saved under the same name. From this point, a directory called /srv/tftp/pxelinux.cfg is created by the *mkdir –p /srv/tftp/pxelinux.cfg* command. In this directory, the PXE boot files are saved in it. While creating the directory, copying the boot files from the server system is also done. During the copying process, a default boot configuration file is created called */srv/tftp/pxelinux.cfg/default* and this is where the configuration is done.

Copying bootfile

```
sudo cp /usr/lib/syslinux/pxelinux.0
/srv/tftp/pxelinux.cfg
```

Creating the default configuration file /tftpboot/pxelinux.cfg/default

```
LABEL ubuntu
kernel  vmlinuz-2.6.32-33-generic

append root=/dev/nfs initrd=initrd.gz netboot=nfs
nfsroot=10.94.204.2:/srv/tftp/ ip=dhcp rw
```

The label of the OS to be used in the PXE boot is Ubuntu. Kernel is a program that constitutes the central core of a computer operating system [10]. Vmlinuz is the name of the Linux kernel executable. Vmlinuz is a compressed Linux kernel which can be executable and bootable. In this case, the kernel vmlinuz version of the Ubuntu is vmlinuz-2.6.32-33-generic. Initrd, also known as (initial ramdisk), is the process of loading temporary file systems in to the Read Access Memory (RAM) in the booting process in Linux.

According to the servers' boot system, the booting system file is initrd.gz. In order for this process of loading the temporary boot file system to occur, a tool called Initial RAM file sharing (intramfs- tool) needs to be installed. This will be the next installation process to be done.

The correct way of finding this kernel vmlinuz version from the server system is by using this command *uname-r.* This means that the PXE client will be running the same OS structure as the server. After the tftp configuration is done and permissions are set, the service is then restarted to work with the new configuration set up. Tftp is restarted by */etc/init.d/tftpd-hpa restart* . The results of the configurations of the tftp files can be found in appendix 2.

While the DHCP server and TFTP server are running perfectly, the need of a nfs-kernel server, initramfs-tools and syslinux come in place. Syslinux in this case is a collection of lightweight boot loaders in Linux that are used in network booting through the enabled PXE network card.

On the other hand, intranfs-tools is used in the process of loading the temporary files to the memory in the booting process. Network File Sharing (Nfs)-kernel server is the functionality behind the transportation of the boot files to the client computers. All these tools can be downloaded and installed using one command. *Sudo apt-get install nfs-kernel-server syslinux initramfs-tools* is the command used for the tools.

Network file sharing (NFS) should be the first to configure after the downloading and installation is done. As NFS is installed, it creates a file named */etc/export* and it uses this file to identify what local directories are available for the NFS clients.

In the NFS export file, it is configured to use the `/srv/tftp/pxelinux.cfg` file to store the operating system files that are used in the booting process. This is done by modifying the export file to know the host machine, by adding the IP address of the host (server) and the network's IP address that will be sharing this file. Then the file is allowed to be exported and it is restarted by *sudo exportfs –a* and */etc/init.d/nfs-kernel-server restart*.

Initramfs-tools and syslinux are configured together by loading the OS onto the initial PXE boot file which is */srv/tftp/pxelinux.cfg/default*. After all the installation and configuration are done and the functions are working as required, the testing of the client machines will take place. This means trying to boot the client PCs from the server. The client PCs are configured to perform diskless booting, and for this to happen. Several steps are taken to complete the booting process of the machine. Figure 4 illustrates the steps taken in the diskless booting process.

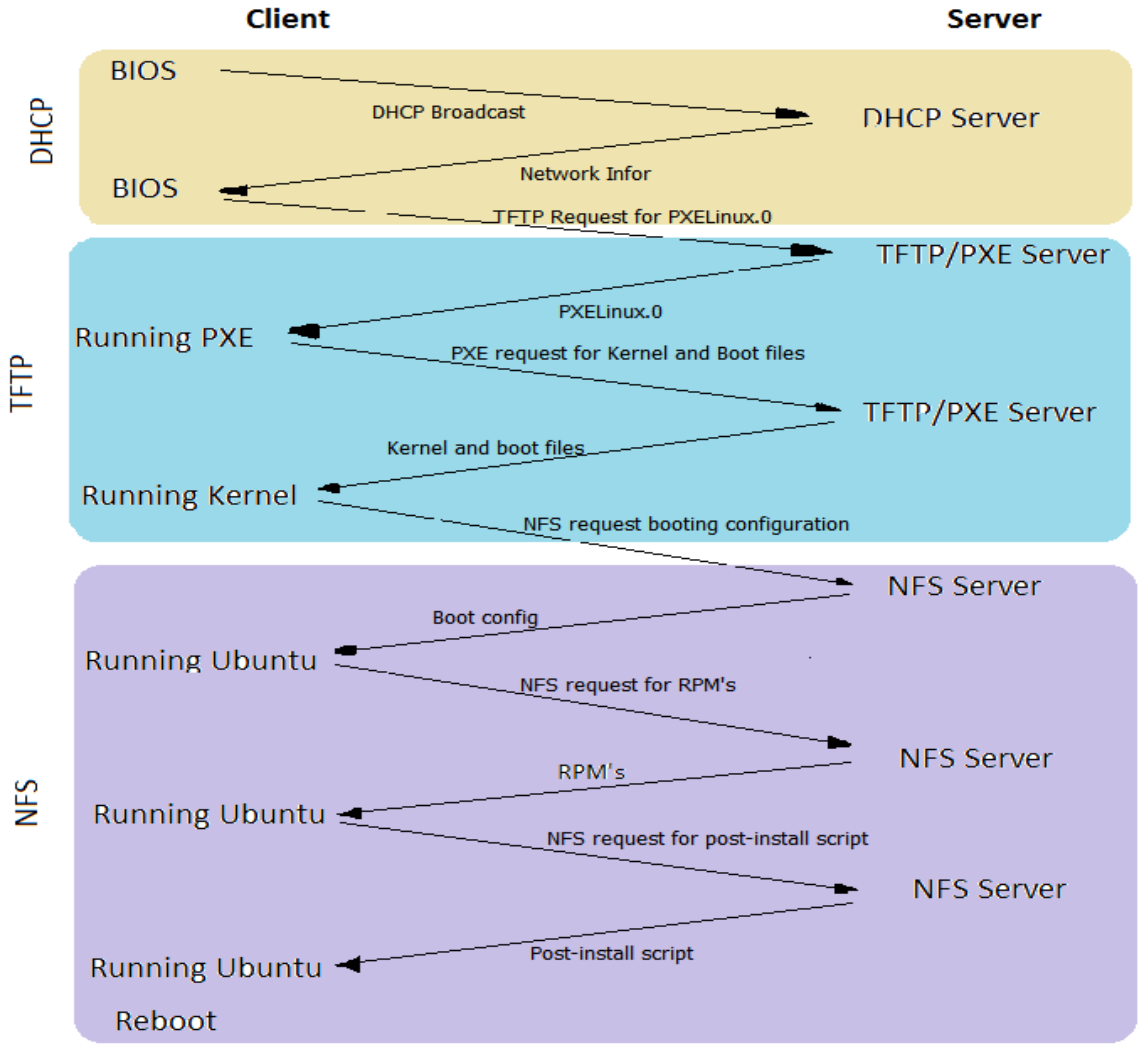

Figure 4. PXE communication process.

At this point, the cluster computers are to be tested for the diskless booting. The first process is to connect them to the power source, connect the network cards using the Ethernet cable to a private switch that is also shared by the server. Through the BOIS setting, network booting is configured. On the switch, it is configured with a single Virtual Local Area network (vlan) and all the interfaces are open.

To add on the switch configuration, fast Ethernet interfaces are enabled to span tree portfast. This feature enhances the switch network reliability, security and enables better management.

Figure 5 illustrates one of the cluster computers performing a diskless booting. It starts by getting the IP addresses from the DHCP server as in step 1 in figure 5. Then from the cluster computer BIOS, it looks for the boot file from its own hard disk. Since there was no operating system on the hard disk, the cluster machine requests for a pxelinux.0 file

system through Trivial File Transfer protocol (TFTP) as in step 2. Since the server was configured with the TFTP server, the file system is delivered to the cluster machine. The cluster machine loads the pxelinux.cfg and while it loads, through Network file System (NFS) step 3, booting configurations are requested. From the NFS server the boot configurations, which are the Vmlinuz and the initrd.gz, are sent to the cluster machine for the booting and running of the Ubuntu Operating system.

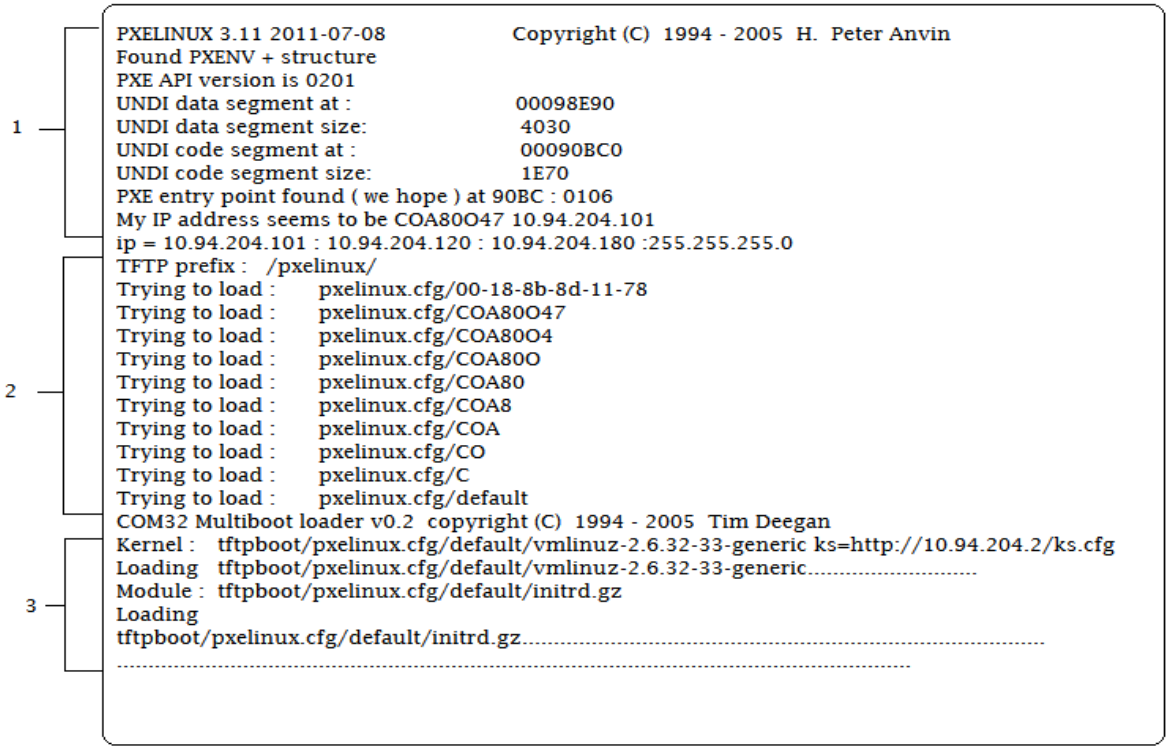

Figure 5. A screenshot of one of the cluster machines performing PXE booting.

The cluster computer will complete loading and installing the boot file on the Read Access Memory (RAM) and then booting the Ubuntu 10.04 operating system. This operation is successful when the Ubuntu logo appears on the cluster screen. The cluster machines are running Ubuntu 10.04 OS. The next process is some post installation and configurations which are done from the server remotely. The OS is then updated after it is up and running. Drivers are installed according to the necessity of them in this project scope. The secure Shell (ssh) client is installed on the machines and a security checkup via ssh is performed as part of the post installation. Then lastly, COMSOL should be installed onto the computers. Due to compatibility issues that occurred while the updating of the OS was being performed, some of the above steps did not go through. Due to this reason, the challenge to continue with this approach increased leading to a stand still on this approach and a chance to try another design to target the aim of this project.

4 UEC and Eucalyptus

The Ubuntu Enterprise Cloud (UEC) is Ubuntu's Eucalyptus powered cloud platform. The Debian Linux distribution Ubuntu has gone to greater lengths to integrate Eucalyptus into their distributions. This means that Eucalyptus and the Ubuntu server can be installed both at the same time. While UEC provides the core function of Eucalyptus, it has some noticeable differences from Eucalyptus. These differences are as follows: UEC adds an image store that allows users with a repository of pre-creating images kept in Ubuntu. Secondly, the registration process is based on zero networking configuration (Zeroconf) implementation that is used in the discovery and in the integrating of components with UEC. Lastly, the UEC color scheme web user interface is designed to confirm with Ubuntu's current theme [11].

Eucalyptus is software based on an open source platform that is available in debian Linux distribution. This helps in creating and managing a private and hybrid cloud. It provides an Elastic Compute Cloud (EC2) compatible cloud platform and a simple storage service (S3) cloud storage platform which is the same as what Amazon Web Service (AWS) does. UEC enables Virtualization Technology (VT) that allows any Ubuntu server to run Kernel Virtual Machine (KVM) as a hypervisor. Hypervisor is a hardware virtualization technique that allows multiple operating systems to run concurrently on a host computer. Hypervisor creates an operating platform for the virtual machines (instances) and provides the execution of this guest operating system.

4.1 Architectural Design

Eucalyptus provides an Infrastructure as a Service (IaaS) type of cloud. This can be either a private or Hybrid cloud. It also provides an interface that lets users to access computer resources available in the network. These resources can be storage, network and processing units. Eucalyptus is designed to have an extensive web service based architecture that enables it to export Application Programming Interfaces (APIs) to the user via client tools. Eucalyptus is now implementing industrial standard Amazon Web Services (AWS) APIs [12]. When Eucalyptus is installed, it comes with some command line tools called Euca2ools.These tools are used by Eucalyptus internally to communicate with Eucalyptus cloud installation including Amazon Elastic Compute Cloud (EC2).The table below shows some features of Eucalyptus in the cloud industry.

Table 4. Features of Eucalyptus in cloud computing.

Features of Eucalyptus
• Compatible with Amazon Web Services • Secure communication with the internal processes • Supports Linux and Windows virtual machines • Supports multiple clusters in single cloud • User and group management • Account reporting and monitoring

As mentioned earlier, Eucalyptus has various components that work together to make the open source cloud work efficiently. For a simple and minimal cloud infrastructure, the components can be divided into two parts. The front end and the node end. The front end comprises of a cloud controller, cluster controller, Walrus storage and storage controller while the node end has one or more node controllers. The functions of these components are described below.

Node controller NC

This is a node computer/server that enables virtualization technology to run Kernel Virtual Machine (KVM) and the hypervisor. KVM is automatically installed when the user chooses to install the node controller at the beginning of the OS installation. The virtual machines (VM) are running on KVM and are controlled by Ubuntu Enterprise Cloud (UEC) and they are known as instances. A node controller runs on each node (computer) and controls the life cycle of the running instances in it. These instances running to the NC have different resources. These resources are determined by the physical resources of this NC, for example the number of core processors, the size of memory and the available disk space. The simple features NC has are like collection of data related to the resources available and reporting to the Cluster Controller and management of the instance life cycle.

Cluster controller (CC)

Cluster controllers manage one or more node controllers and the instances in them. CC manages the network for the instances running on the nodes using different modes of the network mode of Eucalyptus. CC interacts with the node controller on one side and Cloud Controller (CLC) on the other side. The simple feature CC has includes receiving instances requested from CLC and deploying them.

It also checks which NC is available for it to deploy an instance and controls the virtual network for this instance in the NC. Lastly, it collects information about the registered NC and reports to the Cloud Controller.

Walrus Storage Controller (WS3)

WS3 provides a simple storage service (S3) using Representation State Transfer (REST) and Software Of Unknown Pedigree (SOUP) applications which are compatible with Amazon Simple Storage Service. WS3 is a simple file storage system. Some functions WS3 has include storing images of the machines, storing snapshots and serving files using S3 applications.

Storage Controller (SC)

SC provides block storage to be used by the instances. It is almost similar to the Elastic Block Storage (EBS) service used in AWS. It provides storage for snapshots created by Eucalyptus.

Cloud Controller (CLC)

CLC provides a web graphical interface to the user for managing and monitoring the UEC infrastructure on one side and on the other side it interacts with the rest of the Eucalyptus components. Some functions of CLC are monitoring the running instances, deciding which cluster to use for the instances. It also has a comprehensive knowledge on the available resources in the cloud.

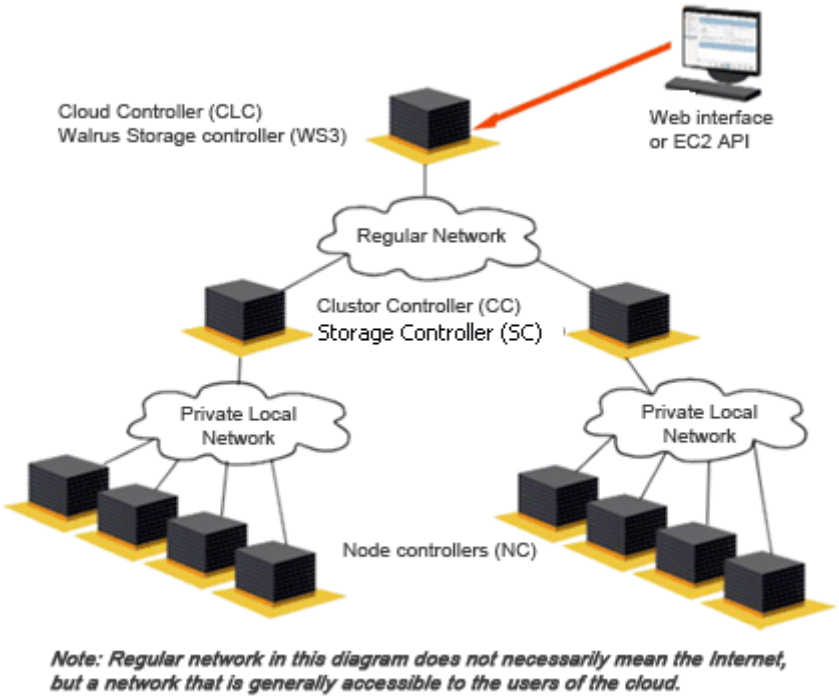

Figure 6. The main components of a simple private Eucalyptus cloud (Adapted from [13]).

As shown in figure 6, CLC is at the top of the front end of the whole cloud infrastructure.CC and SC are connected to the CLC through either regular or private network. Then, the NC's, located at the node's end, are always connected to a private cloud.

4.2 Installation and development

In this topology approach, the idea was to build a private cloud from the resources offered by the IT department of the Metropolia University of Applies Sciences. As mentioned earlier, the resources were the basic 100 desktop computers, a network switch and Ethernet cables. This topology approach was designed and carried out in the same laboratory as the previous topology. There was not enough space in the laboratory for all the 100 computers and so the best option was to use the minimum resource to create a simple working cloud. After this structure works on a small scale topology, it can be transferred to a larger infrastructure. Due to space limitations, three basic computers were used and a network switch was made to connect the computers to form a private network. Figure 7 shows the topology design which was used to build this UEC cloud based on Eucalyptus.

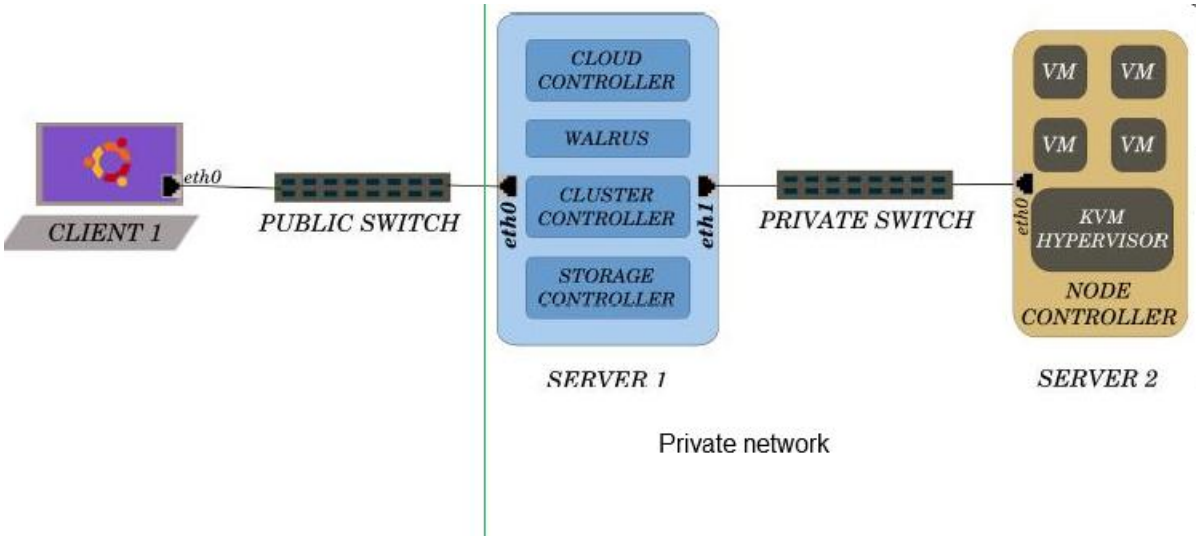

Figure 7. Simple Eucalyptus cloud topology built in the laboratory [14].

Ubuntu Server 10.04 LTE was installed on two of the computers which then became server 1 and server 2 while Ubuntu Desktop 10.04 was installed the client 1 computer. The Ubuntu desktop operating system was installed on to the client 1 computer because of the graphical interface. A browser like Firefox or Internet Explorer was needed for the access of the web interface of UEC to be able to interact with the cloud in the private network. UEC and Eucalyptus have simple installation steps and do not need complex hardware to support the cloud. Table 5 illustrates some hardware suggestions that can be used efficiently with UEC and Eucalyptus.

Table 5. Hardware used for a basic private cloud using Eucalyptus [Adapted from 14].

Hardware	**Server1**		**Server 2**		**Client 1**	
	Minimum	Used	Minimum	Used	Minimum	Used
CPU	1 GHz	2*2 GHz	VT extension	VT,64Bit,Multicore	VT extension	VT,64bit,multicore
Memory	1 GB	2 GB	1 GB	2 GB	1 GB	2 GB
Disk	5400rpm IDE	7200rpm SATA	5400rpm IDE	7200rpm SATA	5400rpm IDE	7200rpm SATA
Disk space	40 GB	200 GB	40 GB	230 GB	40 GB	230 GB
Networking	100Mbp	100Mbp	100Mbps	100Mbps	100Mbps	100Mbps

Installation of server 1

Since the computer does not have any OS, it is booted from a bootable CD that has Ubuntu Server 10.04 64 bit. The steps for the installation in server 1 are illustrated in appendix 6 but a brief explanation of the installation is described in this section. From the booting menu of the bootable CD, Install Ubuntu Enterprise Cloud was selected as the boot option. The installation step continued with region options, language options and time zone menu. As the installation continued, an IP address was required since server 1 has two network interfaces that connect to the Internet on one side and to the node controller on the other interface. The interface facing the public network (Internet) is Eth0 as shown in figure 7. Completing the IP address set up, the next essential part of the installation is the selection of components to be installed in this server. According to the topology in figure 7, the cloud controller, walrus storage, cluster controller, and the storage controller in it are then installed on server 1 machine. All these components are installed by selecting at this point of the installation. The network interface Eth1 facing node controller is configured and it should be the network gateway for the node controller. The next stage is setting up the cluster name to Metrocloud, then selecting the Eucalyptus IP range which was from 10.94.204.100 to 10.94.204.200. When all the steps on installation and configuration are done, the server is restarted to implement the installed items.

The post configuration and installations which are done after the cloud controller is running include setting up a static IP address for the interface facing the node controller, restarting the network and upgrading Eucalyptus. The installation of Network Time Protocol (NTP) package was done after the installation process of cloud controller (CLC). The CLC will act as a NTP server for all the components of the UEC. The time on all components will have to sync with the NTP server to assure the connection of all the components in the UEC network are working correctly. When the UEC network is working correctly, all the components will have the clock source from the NTP server and will show the same time setting. All these configurations are found in appendix 3.

Installation on server 2

The installation of server 2 was the same as of server 1 but crucial minor changes were made in the cloud installation mode. The steps are found in appendix 7. Server 2 is a basic desktop computer with the components given in table 5. The same bootable CD used in server 1 was used in server 2. At the booting menu options, Ubuntu Enterprise Cloud was the installation option for this server.

The installation process continued normally with region options, language options and keyboard layout options. At this point of the installation process, the IP address was set for the network interface. The only interface server2 has is facing the cloud controller. The IP address was set in the same private network as the cloud controller. As the installation proceeds, the Eucalyptus menu of the components is shown.

Since the interfaces are on the same private network, server 2 will automatically recognize there is a cloud controller in the network. Thus server 2 will select itself as a node controller (NC). This is because in every Eucalyptus private cloud, there cannot be two cloud controllers on a single UEC. Confirmation is required for the user installing the OS. The next step is to configure the IP address of the CLC and the network gateway. After the OS is completely installed, the computer is then restarted for it to boot from its own hard disk.

The post installation and configuration on the node controller include updating the OS, upgrading Eucalyptus and installing the of Network Time Protocol (NTP) package. The purpose of this NTP is to synchronize with the NTP server. After installing NTP, the file is then configured to indicate server 1 as the NTP server by the IP address of server 1 eth1 network interface as shown in figure 7. One of the Eucalyptus files in the node controller NC */etc/eucalyptus/eucalyptus.conf* is then configured to fit with the *dhcpd* network, *Pubinterface* and *Privinterface* as *br0*. After all the post configurations, *eucalyptus –nc* is restarted.

For the cluster controller on server 1 to communicate with the node controller on server 2, there is a need for secure communication. SSH public keys are used in this connection. Secure Shell (SSH) is a network protocol used for secure communication between two computer devices. SSH uses (public/ private- key) cryptography to authenticate a remote computer to be allowed to get access into the secured computer. The key is installed into the node controller (NC) for the cluster controller (CC). This is done by creating a tempory Eucalyptus user password on the NC. Then from the cluster controller CC, a command that will copy the SSH identification and the public key from the NC will be set. This is done so that when the cluster controller accessed the node controller remotely, the identity and the public key will be saved on the SSH files of the NC for the authentication process. The command used for this is *sudo –u eucalyptus ssh-copy-id –i eucalyptus/.ssh/id_rsa.pub eucalyptus@10.94.204.10.*

After the identity and keys are copied by trying remote access to the NC by using the password installed on the NC, the password is deleted but the SSH information is saved and cannot be changed anymore.

Installation on the client computer

The purpose of client 1 machine is that it is used by the cloud administrator to configure and interact with the cloud setup. This machine is where the setting up of the virtual machines (instances) and the monitoring of the cloud components is done from. Ubuntu Desktop 10.10 LTE 64 bit OS is installed on this machine and it is considered to be on the public network. It obtains the IP address through the Dynamic Host Configuration Protocol (DHCP) server on the Internet. After the installation of the OS, the Kernel Virtual Machine (KVM) platform is then installed by the `apt-get qemu-kvm` command. Eucalyptus tools called euca2ools are then installed by `apt-get install euca2ools.` These tools are installed on the client computer to be able to manage the cloud. For the client computer to interact with the cloud, a web interface of the cloud controller is used. On the browser address bar, https://10.94.204.2:8443 is the address used to access the CLC web interface. The IP address used is for the CLC eth0 that is facing the public Internet. The default username is admin and the password is changed after the first login. A screenshot of the web interface can be found in figure 8.

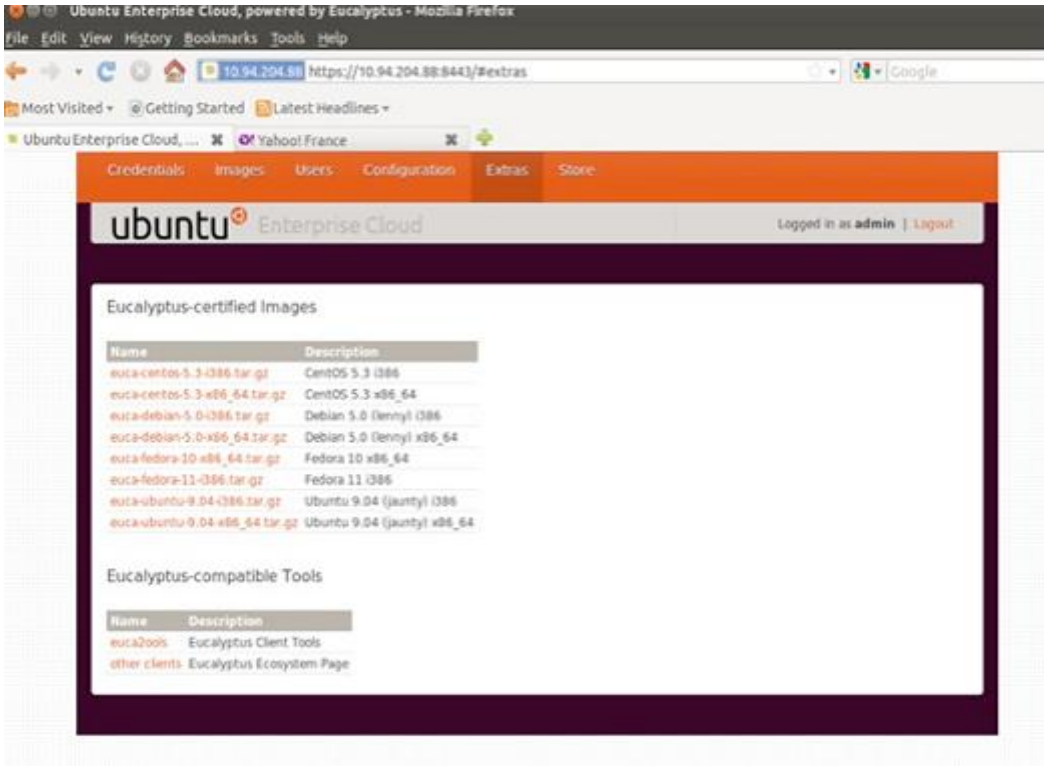

Figure 8. Screenshot of CLC web interface.

After signing in as an admin and providing an email ID for the admin, the next step is to download the credential archives from the credential tab of the web interface and saving them in the *root/.euca* directory on the client computer. It is necessary to extract the credentials archive to get the source *eucarc* script that makes sure the environment variables used by the euca2ools are set into the right place. To verify that euca2ools are communicating with UEC efficiently, the *euca-describe-availability verbose* command is used and the results should show the number of instances the NC can host. Appendix 4 illustrates the results of verifying a command.

4.3 Working on the Instances

The instances in Eucalyptus are created from Eucalyptus Machine Image (EMI). The instances consist of the virtual machines that are used by the cloud user to install any software on the instance and a cloud user can use the instance anywhere through the Internet. EMI is equal to Amazon Machine Image (AMI) used in Amazon. EMIs are combinations of Virtual disk images, Eucalyptus Kernel Image (EKI), Eucalyptus Ramdisk Image (ERI) images and an xml file that contain the metadata about the image [14]. According to this project, Linux OS Ubuntu was installed into the instances. Table 6 illustrates what a Eucalyptus Machine Image (EMI) is made of.

Table 6. An example of an Ubuntu EMI.

XML file Name	Type of Image	ID name in Eucalyptus
Vmlinuz-2.6.28-11-generic.manifest.xml	Kernel Image	EKI – 954313A7
Initrd.img-2.6.28-11-generic.manifest.xml	Ramdisk Image	ERI – D1C61489
Ubuntu-9.04-X86_64.img.generic.xml	Kernel + ramdisk images Eki-954313A7+eri-D1C61247	EMI – 4DC61247

The process of creating an EMI is called bundling. Bundling of this EMI is a process that has multiple steps. Bundling differs between Linux and Windows images. The bundling process is done on the node controller (NC) from the client computer. Table 7 briefly defines the steps taken when bundling an image from the NC.

Table 7. Multi-steps involved in bundling an EMI.

1. Creating a virtual disk image/ space
2. Installing the OS on this virtual image
3. Installing required applications, for example COMSOL
4. Making the OS ready to run under UEC
5. Registering the images with UEC
6. Testing the images

Bundling of a Linux Image

Bundling, as shown in table 7, is to create some disk space where the virtual machine is built on. This means that much space should be allocated for the virtual machines because of elasticity purposes. The disk space is built with kernel virtual machine (KVM) and the command to this is *kvm-img create –f raw image.img 20G* . The more space is allocated, the more time it will take in creating this hard disk space.

Installation of the OS was done and in this project, the virtual machines were running Linux distribution Ubuntu. In the UEC web interface, there was some examples of Linux distro OS. Another option to this can be downloading an Ubuntu ISO image direct from the Internet. In this project, the OS were downloaded from the UEC web interface. Installation is also done by KVM and the command for the installation was *sudo kvm –m 512 /home/metropolia/desktop/euca-ubuntu-9.04-x86_64.iso-drive file=image.img,if=scsi,index=0 –boot d –net nic –net user –nographic –vnc :0* . In the disk space named image.img, which was created in the first step, the ISO image was to be installed and a KVM instance tool was to be booted from it. On the command line above, the –nographic option is given for the virtual machine not to display any graphical output. For any graphical output from the instances, Virtual Network Computing (VNC) is used. VNC is a program installed onto a computer that allows it to share its desktop with a remote computer.

Registering the images to Eucalyptus is the final process in bundling the images. The files that are needed for this image to be uploaded and registered to Eucalyptus are Vmlinuz-2.6.28-11-generic.manifest.xml, Initrd.img-2.6.28-11-generic.manifest.xml and

Image.img.xml. Each file has to be registered in three steps : Euca-bundle-image, Euca-upload-bundle, and euca-register mybucket .

While the registration of these files are taking place, a random identity is produced for each process as shown in table 6 and with that identity is then registered and identified on Eucalyptus with it. Commands on the registering of these files can be found on appendix 5. After all the files have been uploaded and registered on Eucalyptus, a fixed random identity will be given to this new Eucalyptus Machine Image (EMI) and it is the identity used when launching this instance.

Bundling is repeatedly done with different OS ISO images with different applications. Launching and managing of instances is followed after the bundling and registration process is done. Launching of the instances can be done in two different ways: either by using the command line or using graphical tools like add-on found on the Firefox browser. Launching is done from the public network, and in this project, it is done from the client computer. The tool used to manage and launch Eucalyptus instances in this project is Hybridfox. Hybridfox is an open source Mozilla Firefox add-on extension that allows a cloud used to launch, mount volume, map IP addresses and monitor the Eucalyptus instances. Some features of this Hybridfox are illustrated in table 8.

Table 8 Features of Hybridfox [15].

Features of Hybridfox
• List available Eucalyptus machine images (EMI) • List running Instances • Launch new instances • Manage security group and rules • Manage Elastic Block storage (EBS) volume

Installing Hybridfox is done by downloading it from the Google Hybridfox website [15]. The reason for downloading is to be able to open Hybridfox. Launching is done from Tools on the menu bar of Firefox. When running Hybridfox for the first time, selection of the region is done. In the project the region name is Metropolia. After naming the region, another essential process is to feed in the value of Eucalyptus Elastic cloud computing (EC2) _URL which is found from the credentials downloaded on the client computer from the web interface of the cloud controller. Here the credentials were named as euca-credentials-metropolia, and they also come with EC2_access_key and EC2_secret_key which are also found in the credential files. Figure 9 shows a screenshot of the Hybridfox working on the client computer.

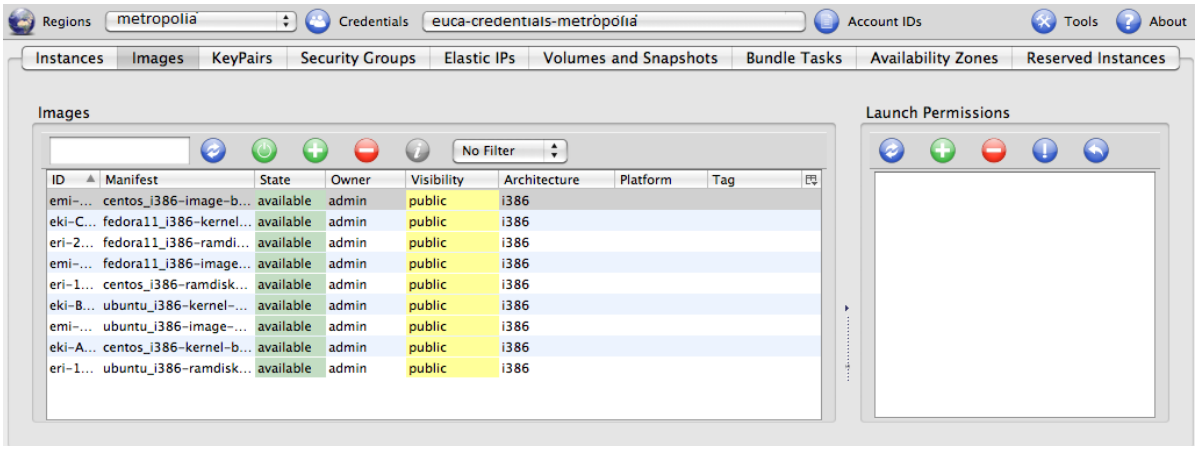

Figure 9. Screenshot of Eucalyptus instances using Hybridfox.

Figure 9 shows the available instances that are ready to be launched by any cloud used. Launching one of the instances through Hybridfox is simpler compared to the command line. On Hybridfox, launching an instance is done by right clicking on the available emi- ID image and it will give options on what size the user would like to use. After selecting the size as shown in figure 10, launch needs to be clicked on and it will appear running in the Hybridfox instances.

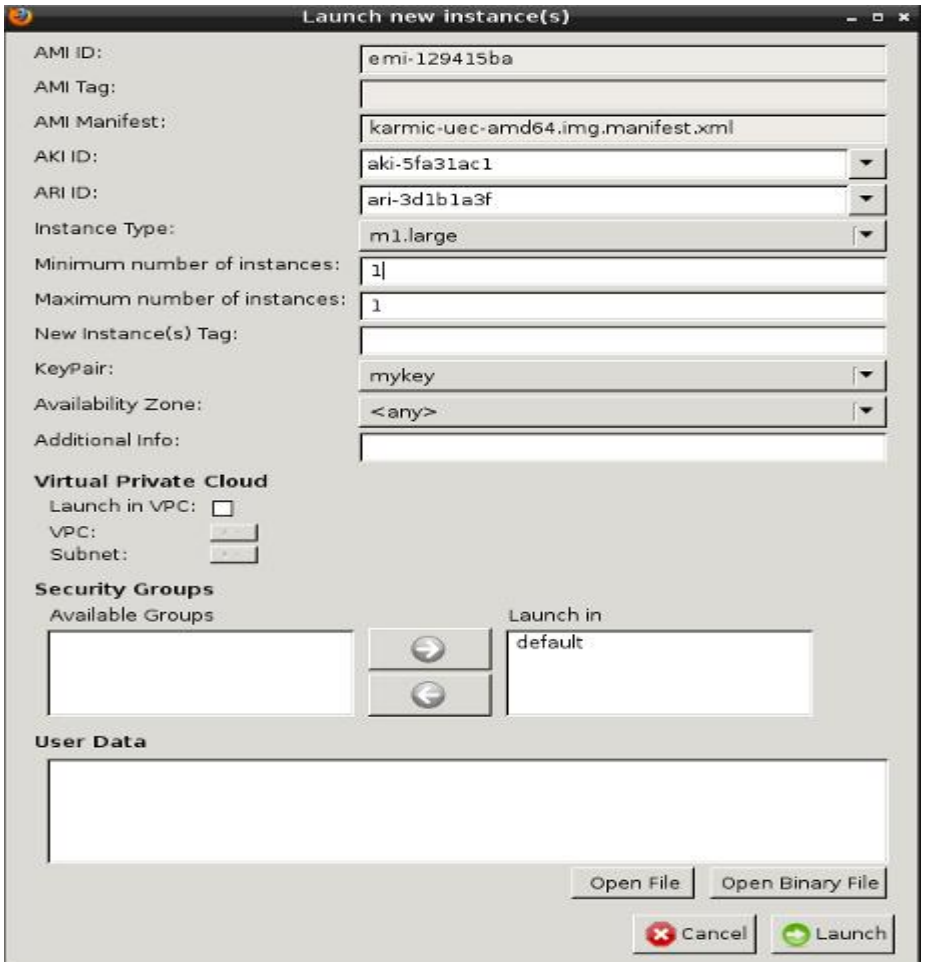

Figure 10. Launching an Instance using Hybridfox.

Launching this instance, the user will be using it as a personal computer through the Internet with a certain application on it. From Hybridfox it will show the IP address of this instance. When using the Ubuntu terminal command line, the user will use Virtual Network Computing (VNC) software and the IP address of the instance to see the graphical interface of this running instance. The command to run the graphical interface of an instance through vnc is *vncviewer 10.94.204.101 :1* where the IP address is for the instance.

At this point the instances are up and running and the end user can use these instances in as many applications installed in these instances as possible. As an administrator, the name of the applications installed in these instances is included in the instance name. For example, if an instance is purposely for the COMSOL software, the end user of the cloud should know where to find it by the name. Using Hybridfox, a user can identify which instance to use and what kind of application it has and he/she can also choose the capacity of the instance.

In this topology of UEC and Eucalyptus as shown in figure 7, the idea of building instances on server 2 worked and the end user on the client computer could see the instances on the node controller. Since the server 2 machine is just a basic desktop computer with normal components as shown in table 5, a command from the client computer is used to check the sizes of the instances available to be used on the single node controller. The output of the command *euca-describe-availability-zones verbose* is shown on table 9.

Table 9. Description of the number and sizes of instances of server 2.

```
. ~/.euca/eucarc
euca-describe-availability-zones verbose
AVAILABILITYZONE   metrocloud                 10.94.204.1
AVAILABILITYZONE   |- vm types      free / max   cpu   ram  disk
AVAILABILITYZONE   |- m1.small      0004 / 0004   1    192     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
```

Table 9 shows the number and sizes of the instances available on server 2. To add more resources for the instances available to the end user of this cloud, an installation of another Node Controller (NC) is done in the same way as in server 2. The new NC is installed in the same network as server 2. After installing the NTP client software, it synchronizes with server 1 and joins the cluster, and the resources change as table 9 shows.

There will be more instances with bigger capacities. Eucalyptus proved to be the best option for the project's Infrastructure as a Service (IaaS) and the next step is to install the COMSOL software into one of the c1.X.large VM types.

Due to minimum time left for my term in this project, the installation of COMSOL was done by a different team and will not be included in this thesis. Since UEC and Eucalyptus were working properly, the next step is to discuss the benefits and drawbacks of using this approach.

5 Benefits and Drawbacks of using Eucalyptus

As discussed and illustrated in the previous chapter, Eucalyptus was successfully used in this project and in this chapter, the benefits and drawbacks to this approach are discussed. The thesis project handles two approaches. The first one, PXE diskless booting, proved not to be successful because of the following reason:

- Too many protocols configured manually
- Much time spend on troubleshooting
- Not enough documentation
- Compatibility problems

On the other hand, UEC and Eucalyptus approach demonstrated some advantages of using this technique for the cloud needed in this project. Some of the advantages are listed below. UEC and Eucalyptus

- are open source.
- come with an Eucalyptus cloud computing API interface.
- compatible with Amazon Web Services (AWS).
- supports public, private and hybrid cloud models.
- come with a management tool designed to build, deploy, and manage one's own cloud.
- provide elasticity and scalability properties in the cloud.
- support Windows and Linux virtual machines.
- secure data communication between the cloud components.
- provides user and group management.
- offer monitoring facility and accounting reports.

Some of the challenges Eucalyptus demonstrated are the following:

- Security – It should be compatible with the institution security policies.
- Client side interface – For the cloud user to be able to use this cloud, he/she should have an idea of working with it, He/she should know how to bring up an instance by command line or using Hybridfox.
- More documentation – For troubleshooting purposes, there is little material about Eucalyptus troubleshooting, but using the Eucalyptus website, solutions can be found on the forums.

While using UEC and Eucalyptus, the merits were greater than the demerits and so the choice became the best option in this project.

6 Discussion

Due to time limitations, I was not able to complete the part of installing the COMSOL software on the UEC and Eucalyptus infrastructure; however, the working condition of this infrastructure could allow any software installation to be done on it. On the other hand, with extensive research on building and managing of an Infrastructure As A Service (IaaS), I was able to understand clearly what hardware, network and application are required when building a cloud.

The aim of the project was to find out if it is possible to build a cloud computing environment from normal desktop computers at the Metropolia University of Applied Sciences . This goal was achieved through the second approach which was Ubuntu Enterprise Cloud (UEC) and Eucalyptus topology. The approach proved to support the hardware provided for the project. It was also noticed that the hardware is elastic and scalable to whatever size required according to the demand of this cloud.

The success of Eucalyptus working with the hardware at hand also opened a door for the development of applications that make the cloud mobile. Eucalyptus lays a foundation for applications that will make this cloud also interact with mobile devices. This kind of application could be designed after Eucalyptus is extracted on a large scale.

Open source cloud computing provides a large avenue of innovation and according to the findings of this project, better designs in large scale can be created and more technical challenges are better dealt with. A strong conclusion on this project is that it is possible to build an open source cloud computing environment that can host multiple applications, expand the cloud on demand and also provide storage to a cloud user and to be easily accessed through the Internet.

7 Conclusion

This project was aimed at finding out how to build a cloud infrastructure using open source software and what kind of resources are needed in building the cloud. In the project, different approaches were used. UEC and Eucalyptus proved to be successful in fulfilling the goal of the project.

The project results proved that it is possible to use normal desktop computers (nodes) linked together to build a cluster. The cluster works with the cloud controller being able to form a Ubuntu Enterprise Cloud (UEC) that can support any operating system and any software application that can be installed into a computer or server. As a cloud developer, it is possible to monitor and also expand the cloud environment using the Eucalyptus frame work based on UEC.

Though the COMSOL software was not installed into the instances, I was able to get the basic and advanced knowledge of three cloud computing models, Infrastructure as a Service, Platform as a service and also Software as a Service. At this point in this project, I was able to conclude that with the basic knowledge of Linux distributions and networking, it is possible for a cloud developer to exploit the full potential of open source software in cloud computing that can be scalable, elastic and reliable to the demand.

8 References

1. Kaufman M, Hurwitz J, Bloor R, Halpe F. Cloud computing for dummies. New Jersey: Wiley publishing, Inc; 2010.

2. Cloud Computing. Cloud computing companies [online]. 2012. URL:http://www.yescloudcomputing.com/cloud-computing-companies/. Accessed 15th January 2012.

3. Krutz Ronald L., Vines Dean Russell, editor. Cloud security: A comprehensive guide to secures cloud computing: Wiley Publishing, Inc; 2010.

4. Furht Borko, Escalante Armando, editors. Handbook of cloud computing. London: Springer; 2010.

5. Hurwitz Judith, Bloor Robin, Halper Fern. Cloud computing delivery models [online]. 2011. URL:http://www.dummies.com/how-to/content/cloud-computing-delivery-models.html. Accessed 17th January 2012.

6. Kavanagh Paul, editor. Open source software: Implementation and management. Elsevier Digital Press; 2004.

7. Kegel Dan. Remote network boot via PXE [online]. 27th October 2002. URL: http://www.kegel.com/linux/pxe.html. Accessed 20th January 2012.

8. TechTarget. Preboot Execution Environment (PXE) [online]. March 2000. URL:http://searchnetworking.techtarget.com/definition/Preboot-Execution-Environment. Accessed 24th January 2012.

9. Introduction to COMSOL Multiphysics 4: Building a model [online]. 2011. URL:http://www.comsol.com/products/tutuorials/introduction/. Accessed 25th January 2012.

10. Bellevue Linux. Vmlinuz definition [online]. 29th March 2005. URL: http://www.linfo.org/vmlinuz.html. Accessed 25th January 2012.

11. Eucalyptus community. The Ubuntu enterprise cloud and Eucalyptus: Eucalyptus and Ubuntu [online]. 14thJanuary2010. URL:http://open.eucalyptus.com/forum /ubuntu-enterprise-cloud-uec-eucalyptus-and-ubuntu. Accessed 29th January 2012.

12. Eucalyptus system. Enterprise cloud, on-premise enterprise cloud IaaS [online]. 2011. URL: http://www.eucalyptus.com/products/eee. Accessed 1st February 2012.

13. Cecaro Fabia. Ubuntu enterprise cloud canonical online virtual training [online]. 14 December 2009 URL :http://blog.vmengine.net/2009/12/14/ubuntu-enterprise-cloud-canonical-online-virtual-training/. Accessed 6th February 2012.

14. Girikumar Yogesh. Eucalyptus beginner's guide – UEC edition [PDF]. May 2010. URL: http://open.eucalyptus.com/participate/wiki/private-cloud-computing-comprehensive-beginners-guide. Accessed 10th January 2012.

15. Hybridfox ,Compute cloud [online]. 2011. URL : http://code.google.com/p/hybridfox/. Accessed 28th January 2012.

Appendix 1

Dhcp server configuration

Nano /etc/dhcp3/dhcpd.conf

```
GNU nano 2.2.2          File: /etc/dhcp3/dhcpd.conf


#option domain-name "";
option domain-name-servers cluster.metropolia.fi;

default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 10.94.204.0 netmask 255.255.255.0 {

range 10.94.204.100 10.94.204.201;
option subnet-mask 255.255.255.0;
option broadcast-address 10.94.204.255;
option routers 10.94.204.99;
filename "pxelinux.0";
}
hostpxe_client  {
hardwareethernet 00:18:8b:8d:11:78;
fixed-address 10.94.204.99;

}
next-server 10.94.204.99;
filename "pxelinux.0";


allowbootp;
allow booting;
#  next-server tftp1.dev.local;
#  use-host-decl-names on;
if substring (option vendor-class-identifier, 0, 9) =
"PXEClient" {
filename "pxelinux.0";
  }
}


# A slightly different configuration for an internal subnet.
subnet 10.94.204.0 netmask 255.255.255.0 {
range 10.94.204.100 10.94.204.201;
option domain-name-servers cluster.metropolia.fi;
option domain-name 10.94.204.99;
option routers 10.94.204.99;
option broadcast-address 10.94.204.255;
default-lease-time 600;
max-lease-time 7200;
```

Appendix 2

Tftp-hpa configuration files

```
  GNU nano 2.2.2       File: /srv/tftp/pxelinux.cfg/default

include mybootmenu.cfg
default ubuntu-installer/i386/boot-screens/vesamenu.c32
Label Ubuntu

kernel  vmlinuz-2.6.32-33-generic
append root=/dev/nfs initrd=initrd.gz netboot=nfs
nfsroot=10.94.204.2:/srv/tftp/ ip=dhcp rw

prompt 0
timeout 100

#/etc/default/tftp-hpa

TFTP_USERNAME=”tftp”

TFTP_DIRECTORY=”/srv/tftp”

TFTP_ADDRESS=”0.0.0.0:69”

TFTP_OPTION=”--secure”
```

Appendix 3

Network Configuration of the cloud controller

```
 GNU nano 2.2.2         File: /etc/network/interfaces
          #auto lo
          #iface lo inet loopback
          iface eth1 inet static
          address 10.94.204.2
          netmask 255.255.255.0
          network 10.94.204.0
          broadcast 10.94.204.255
          gateway 10.94.204.1
root@server1:~$sudo /etc/init.d/networking restart
root@server1:~$sudo apt-get update
root@server1:~$sudo apt-get upgrade eucalyptus
root@server1:~$sudo apt-get install ntp
root@server1:~$nano /etc/ntp.conf
          server 127.127.1.0
          fudge 127.127.1.0 stratum 10
root@server1:~$sudo /etc/init.d/ntp restart
root@server1:~$sudo restart eucalyptus-cc CLEAN=1
```

Appendix 4

```
Root@client1: euca-describe-availability verbose

$ euca-describe-availability-zones verbose
AVAILABILITYZONE metrocloud 10.94.204.10
AVAILABILITYZONE j-  vm types  free / max cpu ram disk
AVAILABILITYZONE j-  m1.small  0002 / 0002 1  192  2
AVAILABILITYZONE j-  c1.medium 0002 / 0002 1  256  5
AVAILABILITYZONE j-  m1.large  0001 / 0001 2  512  10
AVAILABILITYZONE j-  m1.xlarge 0001 / 0001 2  1024 20
AVAILABILITYZONE j- c1.xlarge 0000 / 0000 4  2048 20
```

Appendix 5

Registering kernel image

```
Root@client1:~$ euca-bundle-image -i Vmlinuz-2.6.28-11-
generic.manifest.xml
--kernel true
Root@client1:~$ euca-upload-bundle -b mybucket -m
/tmp/Vmlinuz-2.6.28-11-generic.manifest.xml
Root@client1:~$ euca-register
mybucket/Vmlinuz-2.6.28-11-generic.manifest.xml
```

Registering ramdisk image

```
Root@client1:~$ euca-bundle-image -i Initrd.img-2.6.28-11-
generic.manifest.xml
--ramdisk true
Root@client1:~$ euca-upload-bundle -b mybucket -m
/tmp/Initrd.img-2.6.28-11-generic.manifest.xml
Root@client1:~$ euca-register
mybucket/Initrd.img-2.6.28-11-generic.manifest.xml
```

Registering disk image

```
Root@client1:~$ euca-bundle-image -i image.img --kernel eki-
954313A7
--ramdisk eri- D1C61489
Root@client1:~$ euca-upload-bundle -b mybucket -m
/tmp/image.img.manifest.xml
Root@client1:~$ euca-register mybucket/image.img.manifest.xml
```

Image Listing

```
Root@client1:~$ euca-describe-images
IMAGE emi-4DC61247 mybucket/image.img.manifest.xml

admin available public x86_64 machine
IMAGE eri-A2BE13EC
mybucket/Initrd.img-2.6.28-11-generic.manifest.xml admin
available
public x86_64 ramdisk
IMAGE eki□685F1306 mybucket/vmlinuz□2.6.35□22
```

Appendix 6

Steps of installation of server 1 in UEC and Eucalyptus

- Boot the server off the Ubuntu Server 10.04 CD. At the graphical boot menu, select "Install Ubuntu Enterprise Cloud" and proceed with the basic installation steps.
- Installation allows one to set up the IP address details for one interface. In this project eth0 which is facing the Public network is set as the IP addresses.
- Choose certain configuration options for the UEC, during the course of the install.
- Cloud controller address – Leave this blank as server 1 is the cloud controller in this setup.
- Cloud Installation Mode – Select 'Cloud controller', 'Walrus storage service', 'Cluster controller' and 'Storage controller'.
- Network interface for communication with nodes – eth1.
- Eucalyptus cluster name – Metrocloud.
- Eucalyptus IP range – 10.94.204.100 – 10.94.204.200

Appendix 7

Steps of installation of server 2 in UEC and Eucalyptus

- Boot the server off the Ubuntu Server 10.04 CD. At the graphical boot menu, select "Install Ubuntu Enterprise Cloud" and proceed with the basic installation steps.
- Installation allows one to set up the IP address for one interface. Eth0 is set with a private IP – 192.168.20.2
- The need to choose certain configuration options for your UEC, during the course of the install.
- Cloud Controller Address – 10.94.204.2.
- Cloud Installation Mode – Select 'Node Controller'.
- Gateway – 10.94.204.2 (IP of the CC).