DESIGN AND SIMULATION STUDY OF A SCALABLE AND RESILIENT NETWORK ARCHITECTURE FOR A HYBRID CLOUD; ITS BILLING SYSTEM

A Thesis presented to

AFRICAN UNIVERSITY OF SCIENCE AND TECHNOLOGY (AUST)

Abuja-Nigeria

In Partial Fulfillment of the Requirements

For the award of a

MASTER OF SCIENCE (MSc.) DEGREE IN COMPUTER SCIENCE AND ENGINEERING

BY

NGOLAH KENNETH TIM

SUPERVISOR: Dr. Ekpe Okorafor

December 2011

DESIGN AND SIMULATION STUDY OF A SCALABLE AND RESILIENT NETWORK ARCHITECTURE FOR A HYBRID CLOUD; ITS BILLING SYSTEM

A Thesis presented to

AFRICAN UNIVERSITY OF SCIENCE AND TECHNOLOGY (AUST)

Abuja-Nigeria

In Partial Fulfillment of the Requirements

For the award of a

MASTER OF SCIENCE (MSc.) DEGREE IN COMPUTER SCIENCE AND ENGINEERING

BY

NGOLAH KENNETH TIM

SUPERVISOR: Dr. Ekpe Okorafor

December 2011

DESIGN AND SIMULATION STUDY OF A SCALABLE AND RESILIENT NETWORK ARCHITECTURE FOR A HYBRID CLOUD; ITS BILLING SYSTEM

By

Ngolah Kenneth Tim

A THESIS APPROVED BY THE COMPUTER SCIENCE DEPARTMENT

RECOMMENDED:

_____

Dr. Ekpe Okorafor (Texas A & M, USA)

_____

Prof. Amos David (Nancy 2, France)

_____

Dr. M. Hamada (Aizu, Japan)

APPROVED:

_____

Chief Academic Officer

_____

Date

KEY WORDS

- Scalability
- Pay as you go
- On demand
- Hybrid cloud computing
- Design and simulation
- Fat-Tree-Topology
- Pricing mechanism
- Virtualization

DEDICATION

This thesis work is dedicated to my late father, Bobe Martin Ngolah and my mother Nawain Elisia Funkuin.

.

ACKNOWLEDGEMENTS

I am indebted to many people who have contributed to the success of this thesis. I want to apologize to those whose names will not appear here because of space limitation.

First, I wish to thank God for the grace, peace and protection I enjoyed during my entire stay in AUST, Nigeria.

I wish to express gratitude to my supervisor, Dr. Ekpe Okorafor for making me realize this wonderful area and for his relentless support towards the realization of this work. Words cannot quantify my appreciation for him.

A million dollar thanks goes to the African University of Science and Technology Abuja-Nigeria for granting me a scholarship to study Computer Science and Engineering at MSc. level in AUST. Let me cease this opportunity to appreciate the caliber of professors from different top universities in the world that teach in AUST.

A dozen thanks goes to all of my colleagues in the department of Computer Science and Engineering for their support especially my thesis mate P. Mensah and my friend Mohammad Hassan not leaving out Epie Godfred, Ignace Djitog, Tabot, Hamzat, Grace, Segun, Solomon, Yuru, Olamide, Osa and Larry. I wish you all the best in your future endeavours.

A special thanks goes to the entire Cameroon students in AUST (Vita, Tabot, Edwin, Patrice, Elvis, Polycap, Epie, Edith) for their wonderful support and encouragement during this period.

I wish to thank my “HPC Bringup” team members (Larry, Segun, Mensah, Yaya) for their innovative ideas and the efforts we put together

I want to thank the authors whose works I reviewed and equally to Salako who was very instrumental in the realization of this thesis.

Special gratitude goes to Dr. & Mrs. Ngolah and Ms. Edith Ngolah of Canada as well as the entire Ngolah’s family for their financial and moral support during my stay in Nigeria

In a special way, I want to sincerely thank my wife, Sylvia Ngolah and son, Theron Ngolah, for the patience and sacrifice they underwent during the two years period, while I was away for studies.

Last but not the least, special thanks go to Mr. & Mrs. Tim Francis, Mr. & Mrs. Teng George, Mr. & Mrs. Ngolah Francis, Mr. & Mrs. Ngolah Hycenth, Mr. Ngolah Frederick and GOODMANN Inc. for the support they provided to my family while I was away. I am grateful to you all

Ngolah Kenneth Tim,

Abuja, Nigeria

December 2011

ABSTRACT

A Hybrid Cloud refers to a cloud computing environment that provides pay-as-you-go services to users using the integration of public and private clouds resources. Cloud computing is one of the latest areas in the field of Information Technology (IT) with a promising future. There are predictions that in no distant future, many users will simply go in for cloud computing services as the users will only "Pay-as-they-go" without incurring the cost of the equipments they use or the cost of their maintenance. However, prospective users or customers of the cloud especially those of the hybrid cloud in particular have a phobia for cloud computing services especially in the domain of the billing system since hybrid cloud providers evaluate the bills at the providers' end without the user or customer involved in the calculation.

In this thesis we try to provide a solution for the above mentioned billing problem to the prospective cloud users by coming out with simulations that evaluate the pricing mechanism in a hybrid cloud environment. This evaluation pinpoints the factors that contribute to the total bill presented to the customer and their respective individual contributions depending on whether the computation is data intensive or compute intensive. The simulations were done using the extended cloud simulation software "Cloudsim" which is a simulator that models and simulates cloud computing environment and equally evaluates resource factors. The simulations revealed that the bill incurred by a customer is directly proportional to the degree of intensity of the factor(s) used.

LIST OF ACRONYMS

AUST:	African University of Science and Technology
IaaS:	Infrastructure as a Service
PaaS:	Platform as a Service
SaaS:	Software as a Service
PC:	Personal Computer
OS:	Operating System
VM:	Virtual Machine
EC2:	Elastic Compute Cloud
CRM:	Customer Relationship Management
HPC:	High Performance Computing
DCN:	Data Center Network
VLAN:	Virtual Local Area Network
WAN:	Wide Area Network
MAN:	Metropolitan Area Network
BGP:	Border Gateway Protocol
MIT:	Massachusetts Institute of Technology

IBM:	International Business Machines
OSPF:	Open Shortest Path First
IP:	Internet Protocol
AS:	Autonomous System
DCI:	Data Center Interconnect
MEC:	Multichassis Etherchannel
LAN:	Local Area Network
STP:	Spanning Tree Protocol
DCIN:	Data Center Interconnect Network
RAM:	Random Access Memory
CPU:	Central Processing Unit
MIPS:	Millions of Instructions Per Second
PE:	Processing Elements
JVM:	Java Virtual Machine
CIS:	Cloud Information Service
JDK:	Java Development Kit
JRE:	Java Run time Environment

AMD:	Advance Micro Devices
GHz:	Giga Hertz
L2:	Level 2
GB:	Giga Byte
MB:	Mega Byte
bps:	bits per seconds
SLA:	Service Level Agreement
QoS:	Quality of Service

TABLE OF CONTENTS

CHAPTER TWO – LITERATURE REVIEW

CHAPTER FOUR –SIMULATIONS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

CHAPTER ONE: INTRODUCTION

1.0 GENERAL INTRODUCTION

This section presents the general outline of this work. Chapter one introduces Cloud computing: what it means and how it came about. It also discusses the purpose and the scope of the research work. Chapter two contains the literature review: concepts and the related works. Chapter three presents a feasible methodology for the construction of a scalable network architecture for a hybrid cloud. Simulations to evaluate the pricing mechanism in a hybrid cloud are carried out in chapter four. The work is concluded in chapter five and proposal for future work is presented.

1.1 WHAT IS CLOUD COMPUTING?

Cloud computing is an innovation in the field of Information Technology (IT). It facilitates the way we do computations by making it possible for the use of storage devices, processing devices and other devices by a customer to be as a service rather than the customer incurring the cost of purchase, installation and maintenance of the devices.

Cloud computing is defined by many people [1] in different ways with some of the main vendors- IBM, Amazon, Microsoft, Google, Yahoo and Apple. These vendors are either already providing cloud computing solutions in one form or another or are sponsors of research centers. Although definitions differ in one way or the other, they all center on the same keywords [2]; scalability, on-demand, pay-as-you go, Maintainability etc. this keywords will be elaborated on below. One of the vendors, IBM [1], considers a "cloud" as a pool of virtualized resources that hosts a variety of workloads, allows for a quick scale-out and deployment, provision of virtual machines to physical machines, supports redundancy and self-recovery and could also be

monitored and rebalanced in real time. *Hai Jin* [3], on the other hand sees the "Cloud" as a class of systems that deliver IT resources to remote users as a service. The resources encompass hardware, programming environments and applications. The services provided through cloud systems can be classified into Infrastructure as a service (**IaaS**), Platform as a Service (**PaaS**) and Software as a Service (**SaaS**).

The name "Cloud computing" is a metaphor for the internet. A cloud shape is used to represent internet in network diagrams to hide the flexible topology and to abstract the infrastructure. Some commonly used definitions by cloud community include that of *Ian Foster* [3] who defines cloud computing 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*" and that of *Jeff Kaplan* who views cloud computing as "*a broad array of web-based services aimed at allowing users to obtain a wide range of functional capabilities on a 'pay-as-you-go' basis that previously required tremendous hardware or software investments and professional skills to acquire. Cloud computing is the realization of earlier ideas of utility computing without the technical complexities or complicated deployment worries*". The following are keywords of cloud computing

1.1.1 ON DEMAND

Just like a prerequisite condition, a cloud computing provider needs to satisfy the condition of being able to deliver computing resources to the customers when ever need arises. This means that the cloud provider needs to be able to provide sufficient resources in anticipation of the request from numerous customers. To a customer's advantage, utilizing on-demand resources of

a cloud computing environment reduces the cost incurred in planning, purchasing of equipment as well as the cost of installation. This really eliminates the huge financial burden that should have been borne by the customer. The use of on-demand resources equally avoids having underused resources since the customer may need the use of a particular device only for a certain period of time.

1.1.2 PAY-AS-YOU GO (PAY-PER-USE)

Just like in utility companies where resources like electricity, water or gas are supplied on demand and paid per use, customers in a cloud computing environment pay only for short term use of resources like processor or storage rather than bearing the capital cost of the resources. But unlike utility computing where customers can have access to their physical resources, resources and billing in cloud are found simply in the cloud.

1.1.3 SCALABILITY

To the mind of the customer, the cloud resources are almost infinite, but in reality, the cloud provider has finite resources. In order to solve the issues of scalability, the cloud provider needs to have service level agreement (SLA) with the customer. This will enable the cloud provider to provide the customer's changing needs. This agreement needs to be defined in response time so as to allow the provider to have everything put in place for future demands of the customer.

1.1.4 MAINTENANCE

In a cloud computing environment, maintenance is transparent to the customer. This makes the customer not to worry about issues of repairs and replacements of hardware or software resources as these are done by the cloud provider.

1.2 EVOLUTION OF CLOUD COMPUTING

The field of computing has witnessed technological changes [1] from mainframes computing where many users shared mainframes using dummy terminals to PC computing (stand-alone PCs) where majority of users' needs were solved, to Network computing where there was increase in performance. This trend goes through internet computing where remote applications and resources are utilized, to Grid computing which provided shared computing power and storage through distributed computing system, and finally to the present cloud computing which further provides shared resources on the internet in a scalable and simple way. This can be diagrammatically seen in the following figure.

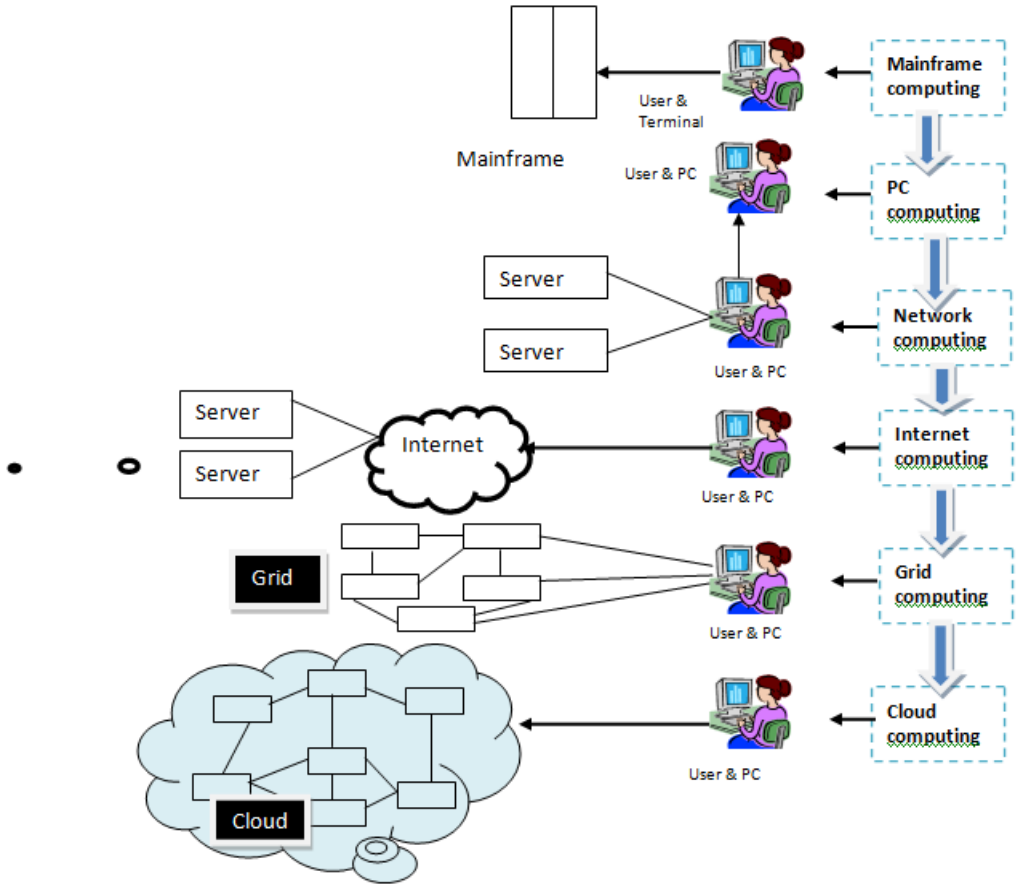


Figure 1. Evolution of cloud computing

All the above mentioned technologies focus on delivering computing power to a large number of end-users in a reliable, efficient and scalable way. Cloud computing, being the latest of the technologies has gained a lot of attention in almost all fields of life: IT, Science, Business *etc.*

1.3 PURPOSE OF THIS RESEARCH WORK

The purpose of this research work is to propose an evaluation mechanism that can be used to evaluate the effect of the billing factors in a scalable and resilient hybrid cloud environment. The evaluation mechanism is used in simulations to ascertain its efficiency in evaluating the billing system, –used to bill the cloud customers. The simulations are done using the extended cloud simulator "Cloudsim [24]" on a single computer node which reduces the burden of looking for a physical data center for the simulation. The main goal of the simulations is to evaluate the effect of the billing factors and that of virtualization in a hybrid cloud.

1.4 SCOPE OF WORK

This research work entails using the best algorithm that will maximize the use of hardware resources like switches in the design of a scalable and resilient Network architecture for a hybrid cloud. To maintain a uniform bandwidth in the entire network architecture, a Fat Tree Topology is used and simulations are done using Java and other complementary software like Apache Ant.

CHAPTER TWO: LITERATURE REVIEW

2.0 INTRODUCTION

This chapter elaborates on the concepts of cloud computing that are relevant to the realization of this research work. It equally presents a review of the works of authors who worked in the related areas.

2.1 CLOUD COMPUTING CONCEPTS

2.1.1 VIRTUALIZATION

For cloud computing to achieve its purpose of scalability, virtualization is used as the underlying technology. Virtualization [4] is the emulation (duplication) of hardware within a software platform. This technology allows a single computer to perform the functions of many computers. Virtualization as defined above is known as full virtualization which allows the simulation of the entire machine. However, there are other different types of virtualizations which do not simulate the entire machine. The most common of which is the *virtual memory* which makes data scattered across RAM and Hard drive to appear as being stored contiguously and in order. Virtualization allows the simulation of hardware via software with the help of virtualization software that is discussed later in this section.

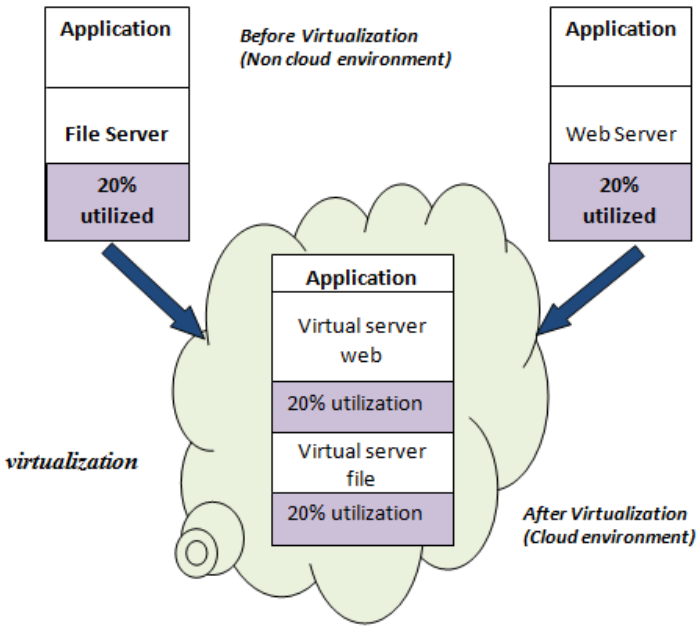


Figure 2.1: Virtualization

As seen in the diagrams above, virtualization helps to:

(a) Increase the use of hardware resources (as the work of two physical servers is done by one server with maximum utilization),

(b) Reduce the management and resource cost (as only one server will be bought instead of two),

(c) Improve business flexibility,

(d) Improve security and reduce run time.

The concept of virtualization came into existence as far back as 1960s [1] where IBM implemented it to help split large mainframe machines into separate "virtual machines". This helped to maximize available mainframe computers efficiency. Virtualization technologies have proven to be of great importance of late. Some of the popular technologies include:

2.1.1.1 VMware

VMware is one of the most widely known virtualization companies. Some of its virtualization applications include:

(a) *VMware WorkStation* which allows users to create multiple x86-based virtual machines on a single physical machine. The majority of these guest operating systems such as Windows, Linux and MAC OSX can be installed onto these virtual machines. This is referred as hosted virtualization application.

(b) *VMware ESX Server*: This does not require a host OS to be installed as it is installed directly onto a server's hardware. This is referred as bare metal virtualization solution or *hypervisor*.

2.1.1.2 APPLICATION VIRTUALIZATION

Application virtualization is the encapsulation of applications from the operation that are installed on. This helps to prevent application malfunction in a case where more than one application use the same drivers. Application virtualization in this case makes separate copies of shared resources and this makes applications to run in complete isolation from one another even though installed on the same operating system.

2.1.1.3 STORAGE VIRTUALIZATION

This involves the virtualization of physical storage devices. This technique allows many different users or applications to access storage, regardless of its location. The virtualized storage appears local to host although it may be distributed across many different locations. An important benefit derived from storage virtualization is the ability of the administrators to mask particular hard drives or storage volumes from particular machines and this improves security and the ability for

the administrator to increase a storage volume in short time especially when the server appears to be running out of space.

2.1.1.4 VIRTUALIZATION SOFTWARE

This is a kind of software that deals with the emulation of hardware and splitting up of both physical and software-based resources. Some of the most popular used virtualization software includes:

a) *Microsoft Virtual PC*: This is Microsoft's hosted virtualization which supports mainly windows operating systems and to a lesser extent some Linux-based operating system.

b) *Sun xVM Virtual Box*: This is sun Microsystems's hosted virtualization software which runs on x86 hardware platforms. It can run multiple virtual machines on windows, Solaris, Linux and Mac OSX.

c) *VMware ESX Server*: This is VMware's bare metal virtualization solution. It is the most widely used solution and can support a wide range of operating systems.

d) Virtual Box: This is a hosted virtualization application created by *Innotek* that can emulate standard x86-based hardware and run a number of different guest operating systems like windows, Solaris, Linux

2.1.2 CLOUD COMPUTING TYPES

2.1.2.1 PUBLIC CLOUD

Public Cloud or external cloud refers to cloud computing that provides services to the general public in a pay-as-you-go manner. The public (customers, individual users or enterprises) access these services over the internet from a third party provider who may equally be sharing computing resources with other customers. Public cloud vendors like Amazon, Microsoft and Google [3] who have vast amount of data centers, enable users to rent their resources at low cost.

2.1.2.2 PRIVATE CLOUD

Private cloud or internal cloud refers to cloud computing on private networks. It is used by organizations to provide services to its customers or users within the fire-wall. Most of the private clouds are enterprises who prefer to keep their data in a more controlled and secured manner.

2.1.2.3 HYBRID CLOUD

Hybrid Cloud which is the main focus of this thesis refers to cloud computing that integrates the activities of public and private clouds. In terms of architecture, a hybrid cloud can be considered as a private cloud that is capable of providing pay-as-you go services to the public (i.e. provides services of the public cloud). *It manages resources in-house and has others provided externally.* In a hybrid cloud, a private cloud is able to maintain high services availability by scaling up their system with externally provisioned resources from a public cloud when there are rapid workload fluctuations or hardware failures. In the hybrid cloud, an enterprise may keep their critical data within their fire-wall and host the less critical ones on the public cloud.

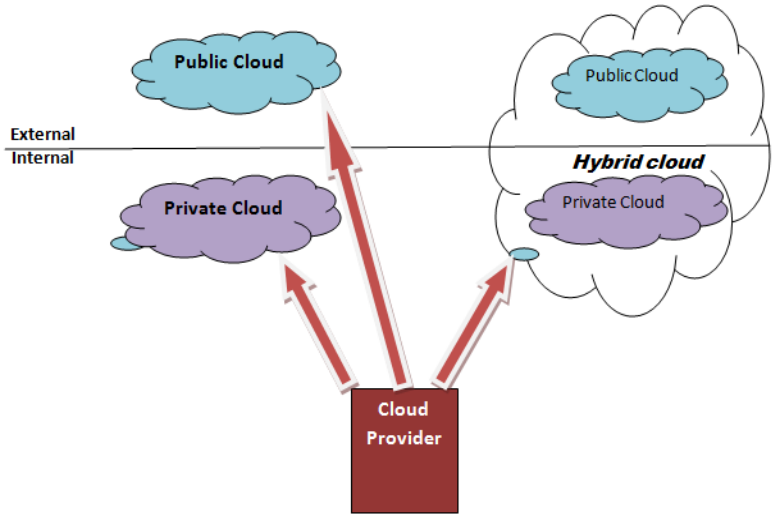


Figure 2.2 Cloud computing types

2.1.3 CLOUD COMPUTING SERVICES

The services provided through the cloud systems to the user as explained below is described in the form of ***XaaS taxonomy*** which was first used by Scott Maxwell [2] in 2006 where 'X' stands for Software, Platform and Infrastructure. Formally the services in the cloud are classified into Infrastructure as a service (IaaS), Platform as a Service (PaaS) and Software as a Service as elaborated on below.

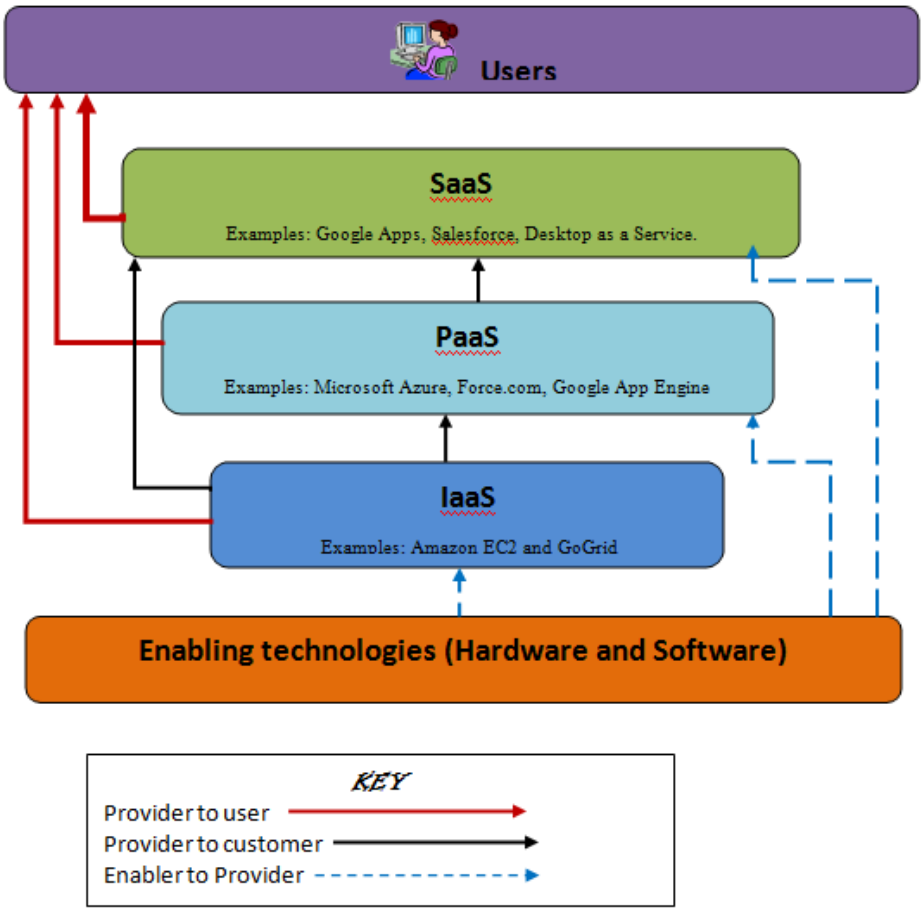


Figure 2.3 Cloud Services (XaaS taxonomy)

2.1.3.1 Infrastructure-as-a-Service (IaaS)

Infrastructure as a service refers to renting raw hardware for use from professional enterprises instead of constructing or buying server. Such services are offered by companies like *Google, Microsoft and IBM* [3]. IaaS offers computation as a Service in which virtual machine based servers are rented for computation. It equally offers data as a Service which unlimited storage space is used to store user's data. The customers of IaaS are not able to control the distribution of the software to a specific hardware platform or change parameters of the underlying

infrastructure. Examples of IaaS include; *Amazon EC2* and *GoGrid*. EC2 (*Elastic Compute Cloud)* provides mature and inexpensive billing system for computing. GoGrid supports multiple operating systems as well as load balancing.

2.1.3.2 Software-as-a-Service (SaaS)

This is the most important service offered in cloud computing. It is based on licensing software use on demand which is already installed and running on a cloud platform. The applications are accessible through a thin client interface like a web browser. These SaaS applications may have been developed and deployed on the Platform-as-a-Service or Infrastructure-as-a-Service layer of a cloud platform as seen in the figure 2.3 above. The customer does not control either the underlying infrastructure or platform. Examples of SaaS applications include *Google Apps*, *Salesforce*, *Desktop as a Service* [3]. Google Apps provide web applications with similar functionality to the traditional office software (like word processing, spreadsheet etc) and also enables users to communicate online with the help of Google mail and Google Talk. *Salesforce* provides business applications such as Customer Relationship Management (CRM) services to consumers like Facebook and Twitter. Desktop as a service provides a virtualized desktop like personal workspace. This helps the users to access their own desktop-on-the-cloud from different places for convenience.

2.1.3.3 Platform-as-a-Service (PaaS)

PaaS represents an intersection between IaaS and SaaS as seen in the fig. 2.3 above. It is a form of SaaS that represents a platform provided to serve as the infrastructure for development and running new applications for the cloud. The environment is the combination of pre-installed operating system integrated with a programming-language-level platform where users can use to

build applications for the platform. The users of PaaS are typically software developers who host their applications on the platform and provide these applications to the end-users. Examples of PaaS include; *Microsoft Azure*, *Force.com*, *Google App Engine* [3]. Google App Engine, maintains programming run time environment on application servers along with some API to access Google service. Microsoft Azure constructs a cloud platform that allows users to move in their applications and manage. Force.com offered by Salesforce, helps service vendors to deliver stable, secure and scalable applications.

The figure below shows the difference in access control right to a customer of Private Infrastructure, IaaS and PaaS.

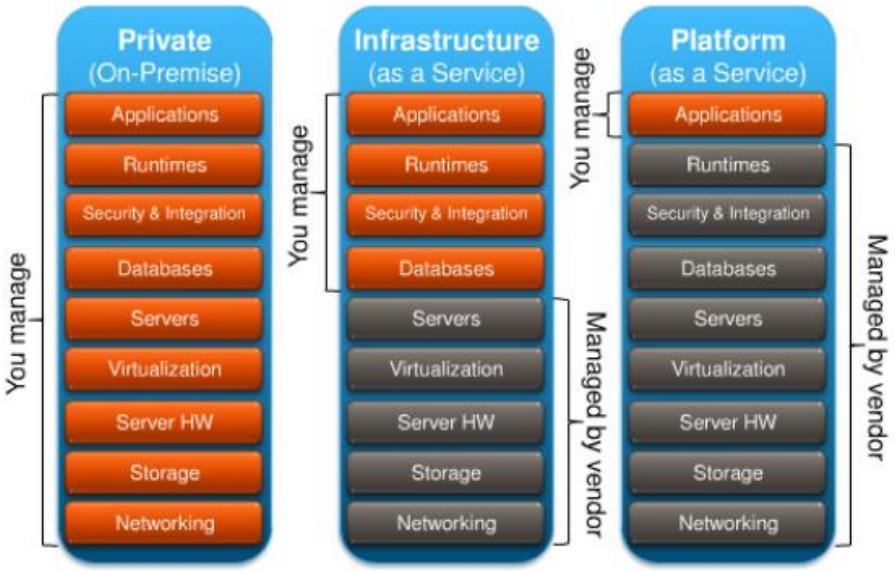


Figure. 2.4: The difference in Customer control right within Private Infrastructure, Iaas and PaaS.

2.1.4 GRID Vs CLOUD COMPUTING

Cloud computing evolved from Grid Computing. In one of the citations [9], Grid computing is described as "*a form of distributed computing where a 'virtual super computer' is composed of a*

cluster of networked, loosely coupled computers acting in a concert" to carry out processor-intensive large tasks. The advantage of cloud over *Grid* [9] is that "cloud computing is not only able to divide large computational tasks into many smaller tasks to run on parallel servers like in Grid, it could also support 'non Grid environment' such as a three-tier web site architecture (i.e separation of presentation, application logic and database)".

Grid computing in essence refers to applying the resources of many computers in a network simultaneously to solve a single problem and is typically used to tackle scientific or technical problems that require a great number of computer processing cycles.

The major difference between Grid and Cloud computing [27] *is that in a Grid environment, a single user can request and consume large fraction of the total resources whereas in cloud environment, individual user's requests are limited to a tiny fraction of the total system capacity because of the scalability feature of cloud.*

Grid computing allows heterogeneity of computers (i.e. computers can have different processor, memory and disk drives) and consist of multiple computers distributed across organizations with the use of Wide Area Network. Communication is with low bandwidth since Grid computing is more of compute intensive than data intensive. Geographically dispersed Grid systems are complicated in management, less reliable and less secured than data-intensive computing systems [7] usually located in secure data center environments. Cloud computing resources on the other hand are often leased from cloud service providers on a pay-as-you go bases. Cloud computing also has the advantage of dynamically provisioning of infrastructure in an elastic and highly scalable manner to match the size of data as Cloud computing is data-intensive. Data-

intensive systems are homogeneous in nature (i.e. computers in computing clusters have identical processor, memory, disk drives with high bandwidth).

Grid's most applications have been noticed to deal with scientific software while software running in cloud concentrates on commercial workload [9]. This can be explained in the fact that Grid middleware is intended for scientific usage while clouds are mostly backup by industries. This leaves Grids with no business opportunities. Cloud computing faces some setbacks like vendor lock-in, security concerns, better monitoring systems etc. most of the technologies developed in Grid are gradually used to accelerate the maturity of the cloud.

2.1.5 GRID, HPC and CLOUD COMPUTING

High Performance Computing (HPC) and Grid Computing are commonly used interchangeably [20]. Grid is concerned with performance as in High Performance Computing. Performance in this context is the capacity of a particular component to provide a certain amount of capacity, throughput or yield of a system or part of a system. Cloud on the other hand is concerned with scalability as in High Scalability Computing. Scalability in this context is the ability of a system to expand to meet demand of a large growing system. Scalability systems may have low performing individual parts. In most cases scalability and performance are orthogonal. Cloud can accommodate Grid and HPC workloads but it is not itself Grid. The figure below shows a trend comparison graph of the number of search citations among Grid Computing, High Performance Computing and Cloud Computing [20].

Figure 2.5 Trends in Search Citations in Grid, HPC and Cloud

Source (Lohr, Google, IBM)

From figure 2.5, the announcement of "Cloud Computing" in October 2007 by Google and IBM [20] leads to the dramatic increase in the trend.

2.2 OVERVIEW OF RELATED WORK

Cloud Computing is a relatively new field that is gradually gaining grounds at a geometric rate. A number of authors have come up with different versions of network architectures and pricing mechanisms: Vahdat et al [17] of University of California, presented a network architectural model that uses commodity switches for the construction of a data center. They argued that the use of commodity switches will support full aggregate bandwidth of cluster in a data center as well as will deliver more performance at a less cost than common available network architectures. Buyya et al [14] of University of Merbourne in Australia, presented a model for the cloud market which focuses on a two-layer design that discusses the metrics used in costing in SaaS providers. Again, Buyya et al [25], have proposed the use of market-oriented resource management in order to regulate the supply and demand of cloud resources at market

equilibrium. They equally proposed the architecture for market-oriented allocation of resources within cloud as seen in the figure below.

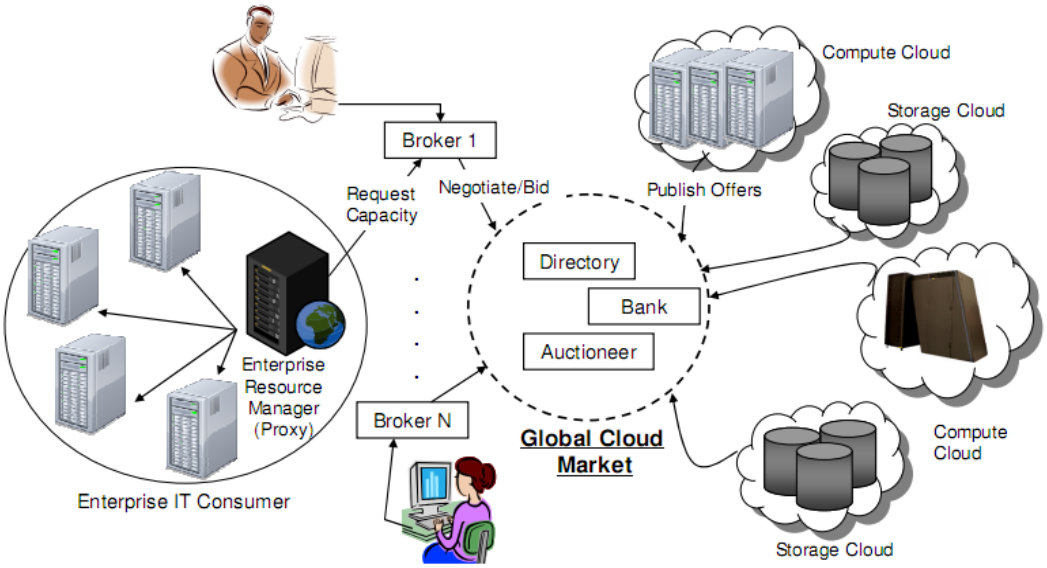


Figure 2.6 Market Oriented cloud computing

Sahoo et al [21] presented a mathematical market-oriented model in which they discuss the analogy between "ant and the bee system" to that of "Virtual machine monitor and the Dispatcher". This model highlights the need for a busy monitor (bee analogy) and the use of the shortest path by the dispatcher (ant analogy) to improve on the market output in a cloud market. None of the above mentioned works have given any attention to the pricing mechanism in a scalable and resilient hybrid cloud.

CHAPTER THREE: HYBRID CLOUD NETWORK ARCHITECTURAL DESIGN

3.0 INTRODUCTION

This chapter discuses the network architectural model of a hybrid cloud with much attention on the mathematical model of the scalable data center. It also discusses the types of routing protocols used at different levels of the design and finally ends with a proposed solution to the security and confidentiality aspects of data in a hybrid cloud.

3.1 HYBRID CLOUD NETWORK DESIGN

The hybrid cloud is constructed by interconnecting the components of a private cloud to those of a public cloud. The constructed architecture then provides services to the immediate users of the organization as well as the public. The components for the architecture include: Data Center Network, Data Center Interconnect Network, Cloud in a box and Network Service Nodes. For the purpose of this thesis, the network architecture of the components will be individually discussed, finally, presenting the "*big picture"*.

3.1.1 DATA CENTER NETWORK DESIGN

Data Center Network Architecture is an architecture that interconnects cloud resources like servers and storage devices in a cloud data center. The data centers in a cloud environment are used for the provisioning of scalable cloud services. This scalability is achieved with the help of the construction of Data Center Network (DCN) that may connect as many as thousands of servers together which then provide scalable services to the end users. The data center is made

up of the core layer, the aggregation layer and the access layer [16]. Below is a view of a common Data Center Network Architecture which uses 10 GigaE Switches at aggregation layer and GigaE switches at access layer.

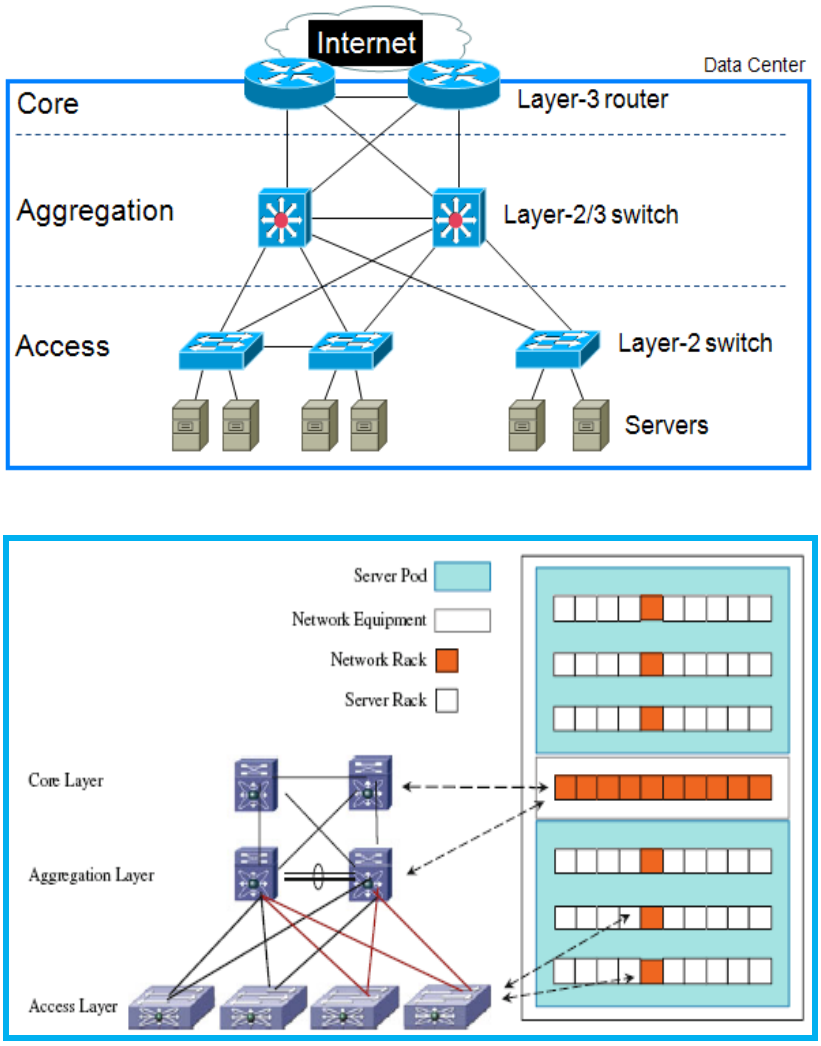


Figure 3.1 Common data center Network Architectures.

From the diagram above, the access layer of the data center is concerned with the servers of the data center. This layer provides resources like switches used to interconnect servers in the data center. There are three most commonly used approaches for server networking in access layer of the data center. The approaches are:

a) *End-of-row switching*: this approach is cost effective in port utilization (like in Cisco catalyst 6500 series switch), and it is also in most cases server-independent. This independency provides flexibility in the use of different servers.

b) *Top-of-rack switching*: in this approach, the switch (e.g Cisco catalyst 4948 switch) is placed at the top of the rack and cables are used to connect this switch to all the servers. This approach is advantageous in that it supports fast port to port switching and a simplified cable management.

c) *Integrated switching*: this approach uses blades which support a small number of servers (e.g Cisco catalyst blade 3000).

At least two of the approaches are used in access layer at any given time. The approaches equally have trade-offs and their use depends on the server specifications.

The aggregation layer provides a meeting point for access layer and core layer. This layer actually serves as a boundary between routed links of layer 3 and Ethernet broadcast domain of layer 2 with the use of 802.1Q VLAN trunks for connection of access layer and aggregation layer. *A VLAN is a logical grouping of networks users and resources connected to administratively defined ports on a switch which allows the creation of smaller broadcast domains within a layer two switch internetwork by assigning different ports on the switch to different subnetworks.*

The core layer is responsible for the provision of layer 3 switching facilities for IP traffic between blocks in the network (e.g campus, internet or P2P) from the data center to the internet backbone. Depending on the geographical location of the data centers, the traffic may be conveyed using private WAN, MAN or public internet. The core of the data center network is typically broken down into high performance, highly available, chassis-based switches. The core

switches are configured with BGP (Border Gateway Protocol) to interact with the WAN. In small data centers with fewer servers, one or two layers can be collapsed.

The main purpose of this section of the thesis is to present a design of a data center network architecture that: is resilient (*the ability to maintain the intended function after a server failure*), is scalable (*able to support millions of servers without affecting the functionality of the data center network*), can provide high bandwidth and should allow incremental expansion. This purpose is achieved by interconnecting managed GigaE switches in fat-tree architecture [16] using a three-tiered design (core, aggregation and access layers). The choice of managed GigaE switches over unmanaged switches in this thesis is that, the managed switches will have the ability to be controlled (i.e. can accept IP address) and will have additional features like Quality of Service (QoS) –which allows the prioritization of network critical traffic like voice. Managed GigaE switches will be cheaper than the corresponding port-routers and will equally support VLANs and redundancy. Fat-tree is a network topology design by Charles E. [29] of MIT. This is a universal network architecture for efficient communication with links getting fatter as one moves up the tree towards the root. The increase in fatness is to alleviate the bandwidth bottle neck closer to the root. Figure 3.2 below shows the structure of a fat tree topology.

Table 3.1: Differences between Common Data center Topology and the Proposed Topology

Common Data Center	**Fat-Tree (Proposed) Topology**
Uses 10 GigaE switches at aggregation layer	Uses GigaE switches in all layers
Average price [28] for 48-port 10 GigaE switch = $18K	Average price for 48-port GigaE switch =$2K
Max No. of host [17] for 32-port 10 GigaE = 7680	Max No. of host for 32-port GigaE= 8192
Max No. of host [17] for 64-port 10 GigaE = 10240	Max No. of host for 64-port GigaE= 65536
Max No. of host [17] for 128-port 10 GigaE = 20480	Max No. of host for 128-port GigaE=524288

This comparative analysis shows that the proposed topology is more scalable and less expensive.

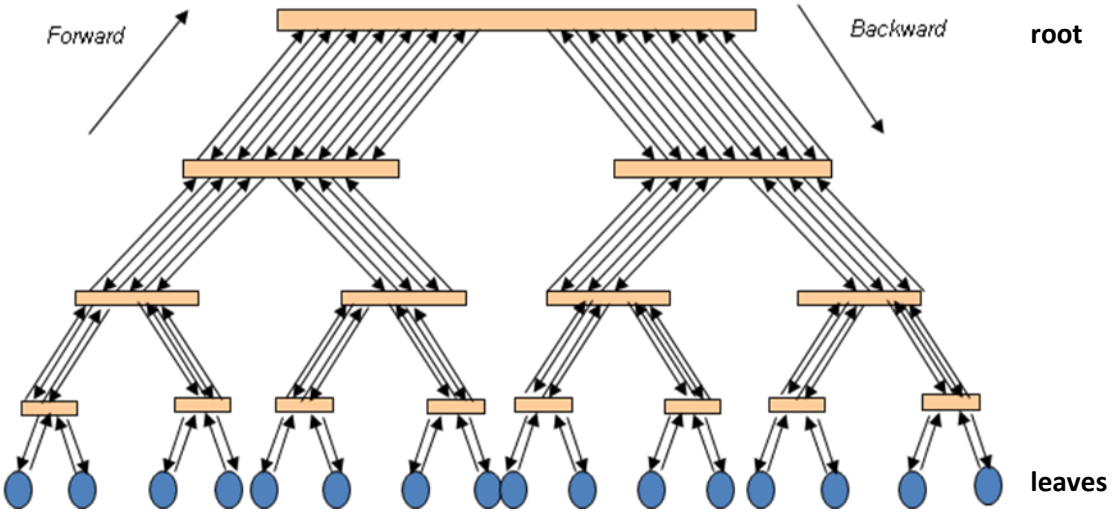


Figure 3.2: Structure of a Fat-tree topology

The choice for the use of fat-tree topology for this thesis is that it supports the use of identical switches and delivers high level of bandwidth to many end devices.

3.1.1.1 MATHEMATICAL MODEL OF THE DCN ARCHITECTURE.

In the fat-tree-topology, the concept of arrays is used. The topology rules states that in a *k*-ary fat-tree, there are k-pods (*A pod consists of elements like aggregation switches, access layer switches, servers and storage within a self-contained box*) with each pod containing two layers of *k/2* switches and every *k*-port switch in the access layer is directly connected to *k/2* servers. Each of the remaining *k*/2 ports of the access layer is connected to *k/2* of the *k* ports in the aggregation layer of the hierarchy. There are $(k/2)^2$ *k*-port core switches each of which has one port connected to each of the *k* pods. The i^{th} port ($i \in [0,k\text{-}1]$) of any core switch is connected to pod *i* such that consecutive ports in the aggregation layer of each pod switch are connected to core switches in *k*/2 ways. A fat-tree built with *k*-port switches support $k^3/4$ servers or hosts. This makes the topology very scalable. The routing protocol that routes IP packets in this DCN

Autonomous System (AS) is Open Shortest Path First (OSPF) which uses bandwidth as its metric of shortest path. There are $(k/2)^2$ paths between any two servers on different pods and only one is chosen.

Assigning IP addresses in the network

Let k belongs to $\mathbb{N}$ (the set of natural numbers), k even,

$0 \leq p \leq k$ -1, p is a pod number in DCN,

$0 \leq s \leq k$ -1, s is a switch position in p from left to right,

$2 \leq ID \leq (k/2)+1$, ID is a server position in the subnet,

i,j are switch coordinates in $(k/2)^2$ switch grid,

Then to maximize the number of host servers in the network, IP addresses assigned are within the class A IP address private block of 10.0.0.0/8 as follows:

10.p.s.1 for pod switches,

10.p.s.ID for servers,

10.*k.j.i* for core switches, ***k*** **< 256**

Simulated model with *k*= 4

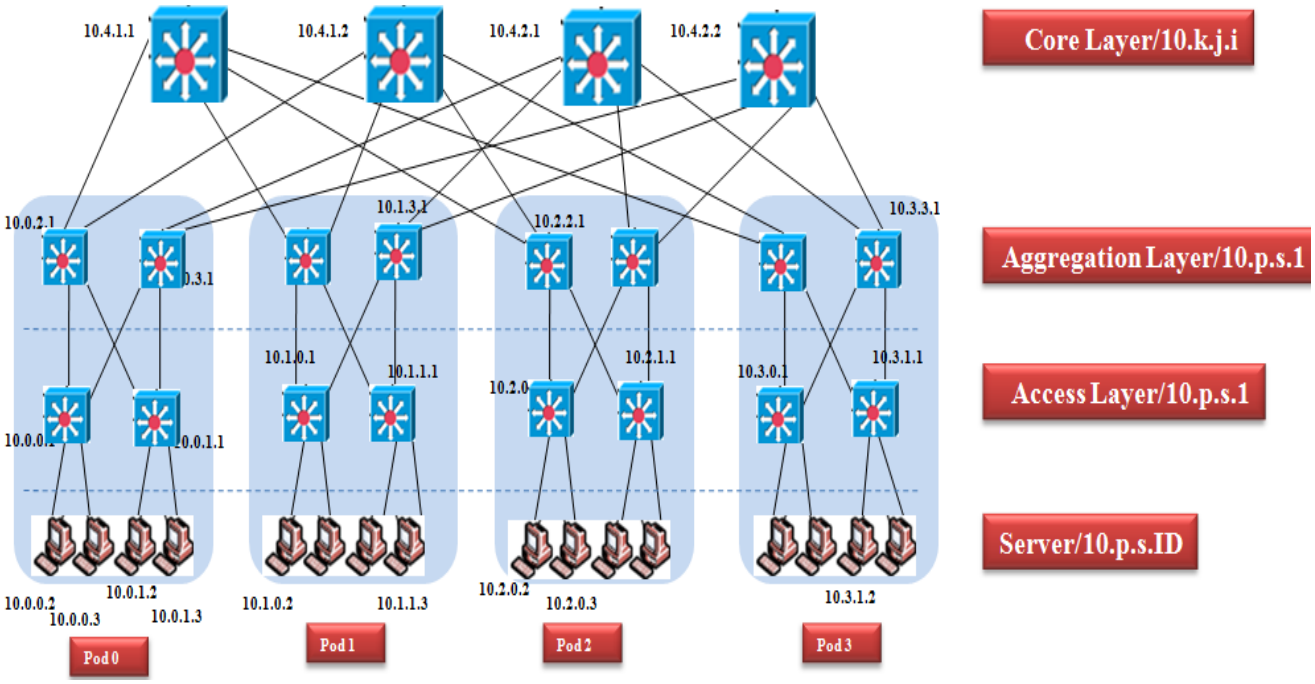


Figure 3.3: Simulated Data Center Network Architecture for k=4

3.1.1.2 ROUTING IN THE TOPOLOGY

As indicated above, OSPF is used as a routing protocol to send IP packets within this autonomous system. The topology uses two level look-ups to send traffic; the first level is prefix look-up which is used to route down the topology to servers and the second level is suffix look-up which is used to route traffic towards the core. The routing table of any pod switch at any time will not contain more than *k*/2 prefixes and *k*/2 suffixes. The first routing lookup has (prefix, port) mapping and left-handed entries (that is: /m prefix masks of form 1^m0^{32-m} , *m* indicates the bits used in the block) and equally points to the second routing table which has (suffix, port) mapping and right-handed entries (that is: /m suffix masks of the form $0^{32-m}1^m$). The number of prefixes or suffixes for any switch in the network is such that the first level prefixes can not be more than *k* and the second level suffixes can not be more than *k*/2. The prefix (first level) is

considered to be terminating if it does not contain any second level suffix. The pod switches have terminating prefixes to the subnet of that particular pod and so if a server sends a packet to another server (host) on a different subnet of the same pod, then all upper layer switches in that pod will have a terminating prefix pointing to the destination subnet switch. For all other packets sent from one pod to another, the pod switches will have a default /0 prefix (i.e non terminating) with secondary table matching server ID. All core switches are assigned terminating first level prefixes for all networks IDs corresponding to the appropriate pod of that network.

3.1.1.3 HOW PACKETS MOVE IN THE TOPOLOGY

Once a packet leaves its destination and reaches a core switch, the core switch will contain only one path to its destination pod and the said core switch will include a terminating /16 prefix for the pod of that packet (i.e 10.pod.0.0/16, port). When the said packet arrives at the destination pod, the receiving upper layer pod switch will include a /24 prefix (i.e 10.pod.switch.0/24, port) to direct the packet to its destination subnet switch where it is finally switched to its destination server.

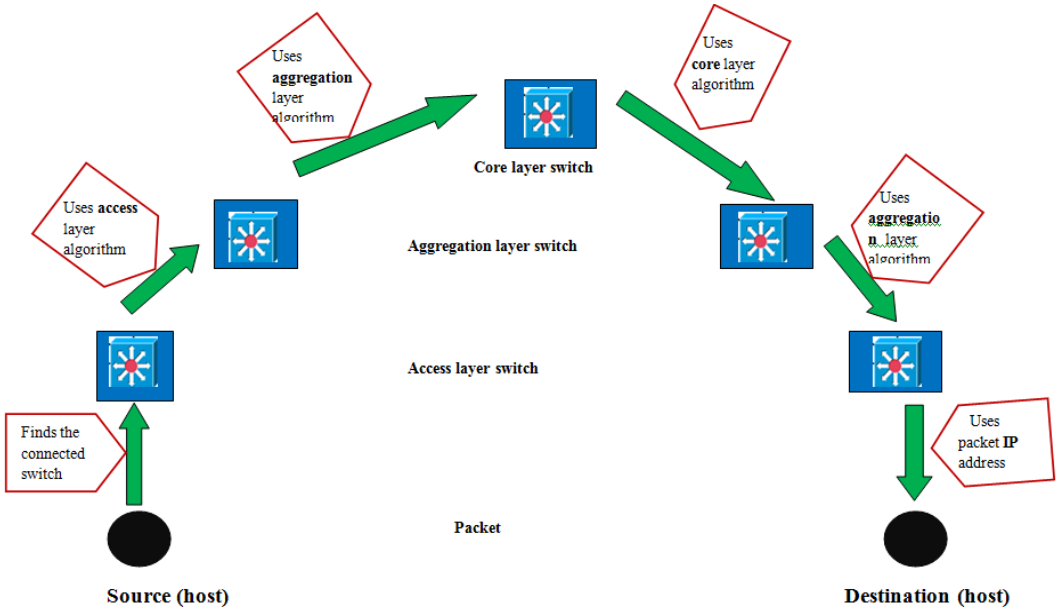


Figure 3. 4: Packet flow diagram in the topology

3.1.1.4 Algorithms for the forwarding table in the topology

a) Algorithm (pseudocode) to generate routing table for switches at the access layer level.

```
foreach pod x in [0, k − 1] do   // a foreach loop is used to loop through an array
        foreach switch z in [0, (k/2) -1] do
                addPrefix(10.x.z.1, 0.0.0.0/0, 0);
                foreach host ID j in [2, (k/2) + 1] do
                        addSuffix(10.x.z.1, 0.0.0.j/8,(j − 2 + z)mod(k/2) + (k/2));
                end
        end
end
```

b) Algorithm (pseudocode) to generate routing table for switches at the aggregation level.

```
foreach pod x in [0, k − 1] do  // a foreach loop is used to loop through an array
        foreach switch z in [(k/2), k − 1] do
                foreach subnet i in [0, (k/2) − 1] do
                        addPrefix(10.x.z.1, 10.x.i.0/24, i);
                end
```

```
            addPrefix(10.x.z.1, 0.0.0.0/0, 0);
            foreach host ID j in ⌈2, (k/2) + 1⌉ do
                addSuffix(10.x.z.1, 0.0.0.j/8,(j − 2 + z)mod(k/2) + (k/2));
            end
        end
    end
```

c) Algorithm (pseudocode) to generate routing tables for switches at the core level.

```
    foreach j in ⌈1, (k/2)⌉ do   // a foreach loop is used to loop through an array
        foreach i in ⌈1, (k/2)⌉ do
            foreach destination pod x in ⌈0, (k/2) + 1⌉ do
                addPrefix(10.k.j.i,10.x.0.0/16, x);
            end
        end
    end
```

The functions in the algorithms are of the form addprefix(switch,prefix,port) and addsuffix(switch,prefix, port) where switch represents the switch address and port represents the output port of the switch.

Using k = 4, source = 10.0.1.2 and destination = 10.3.1.2 in the algorithms above, we have the following routing tables

Prefix	Output port
10.3.0.0/24	0
10.3.1.0/24	1
0.0.0.0/0	

suffix	Output port
0.0.0.2/8	3
0.0.0.3/8	2

Table 3.2: Routing table for aggregation layer for destination 10.3.1.2

Prefix	Output port
0.0.0.0/0	

Suffix	Output port
0.0.0.2/8	3
0.0.0.3/8	2

Table 3.3: Routing table for access layer for destination 10.3.1.2

Prefix	Output port
10.3.0.0/16	3

Table 3.4: Routing table for core layer for destination 10.3.1.2

3.1.1.5 Simulation of the algorithm with k= 4, and sending a packet from source address 10.0.1.2 to destination address 10.3.1.2.

Sending a packet from a source address 10.0.1.2 to destination address 10.3.1.2 will pass through the default gateway switch with address 10.0.1.1 of that particular subnet. The gateway switch will use the access layer algorithm and will match the packet with /0 first level prefix, then it will forward it using the destination server ID byte of the secondary table of that prefix which matches with 0.0.0.2/8 suffix. This suffix points to output port 3 and switch 10.0.3.1. Switch 10.0.3.1 then uses the aggregation algorithm and matches the packet with /0 first level prefix and forwards the packet using the destination server ID byte as above of the secondary table of that prefix which matches with 0.0.0.2/8 suffix. This suffix points to the core switch 10.4.2.2 through output port 3. The core switch uses the core layer algorithm and matches the packet to

10.3.0.0/16 prefix. This prefix points to switch 10.3.3.1 of pod 3 through port 3. This switch (10.3.3.1) uses aggregation layer algorithm and since it belongs to the same pod as the destination subnet, it will have a terminating prefix, 10.3.1.0/24 which points to the switch of that subnet 10.3.1.1 on port 1. This switch finally delivers the packet to the server 10.3.1.2 using layer two switching. This is diagrammatically shown below.

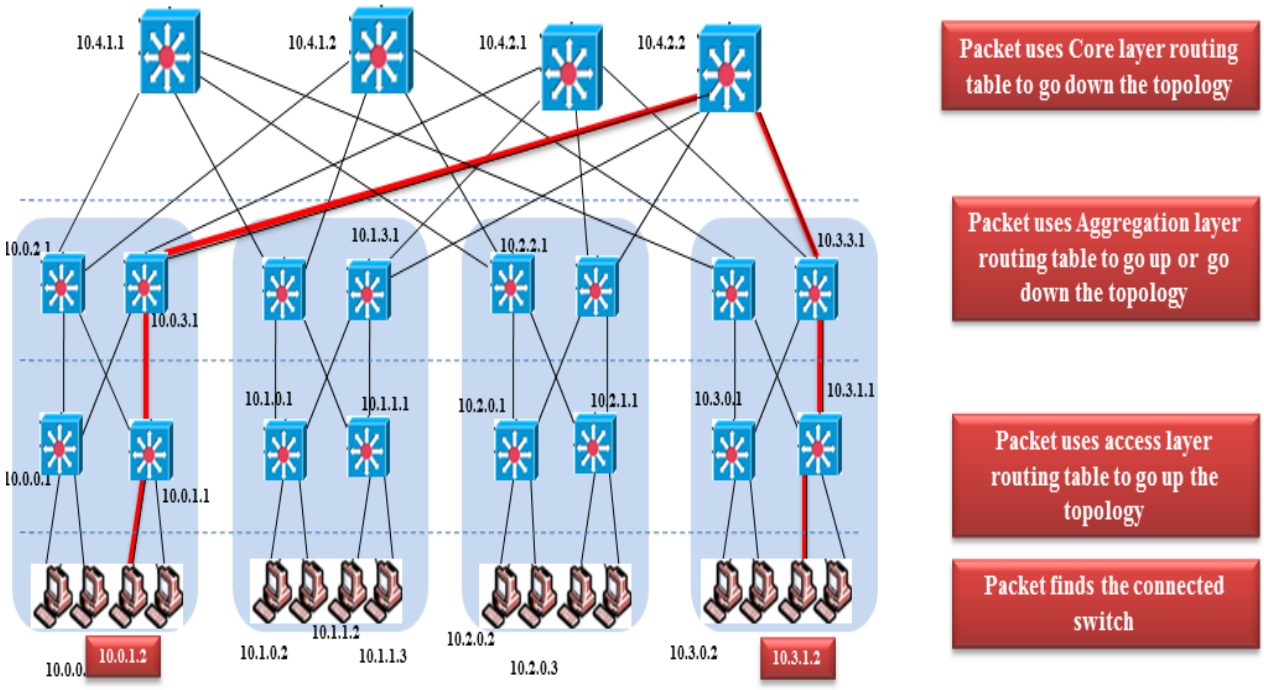


Figure 3.5: Movement of a packet from source to destination in the Data center: The red lines indicate how a packet moves from source 10.0.1.2 to destination 10.3.1.2 using a fat tree topology

A CENTER INTERCONNECT DESIGN

The hybrid Cloud design deals with the interconnection of a Public cloud and a private cloud. To interconnect these two autonomous systems (AS), a Border Gateway Protocol, BGP v4 is used in a Data Center Interconnect (DCI) Network solution which is used to connect multiple data centers together. This extends subnets beyond layer 3 boundaries of a single data center. DCI Network solution [8] helps in building resilience into a system with the use of backup data

centers (fig 3.5 below) and also maximizes a company's server virtualization strategy. This data center resiliency is achieved by deploying dual layer switches with each switch linking to the data center, thereby forming a redundant data center interconnect. This multi-switch links forms part of a Multichassis Etherchannel (MEC)

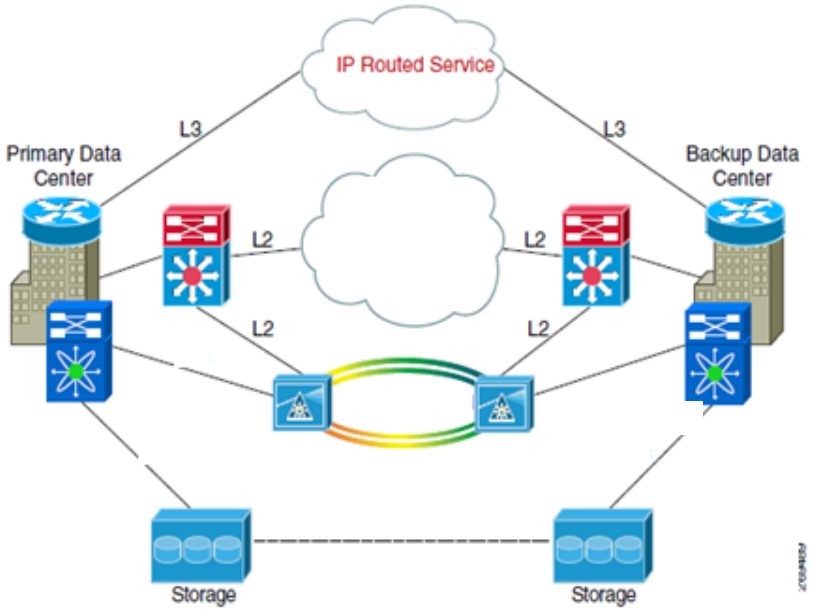


Figure 3.6: Resilient data center

Availability is the prime concern when constructing an interconnect between data centers. This availability is achieved with the use of redundant connections between sites so that the impact of failure of a single link will not be felt. To avoid a loop in the redundant connections, a Spanning Tree Protocol (STP), *(a network protocol that ensures loop-free topology in LAN*) is used to assure that only one active part is in use. The Spanning Tree parameters are configured so that even VLANs are forwarded over one link and odd VLANs are forwarded over the other link to avoid the conflict of all redundant connection links being in use. STP domain spans the whole of VLAN. This makes things easier when multiple data centers share a common VLAN as the STP

domain will then extends to all the data centers in the said VLAN. For the purpose of this project, the target is to interconnect two data centers (one active and the other as backup) each in both private and public clouds using one VLAN that will provide high bandwidth to users.

The design uses multichassis etherchannel to connect a pair of catalyst 6500 switches running on VLAN at the DCI layer of either data center. The links between the VLAN pairs as well as switches to the aggregation layer in each data center are Gigabit Ethernet to allow a uniform bandwidth.

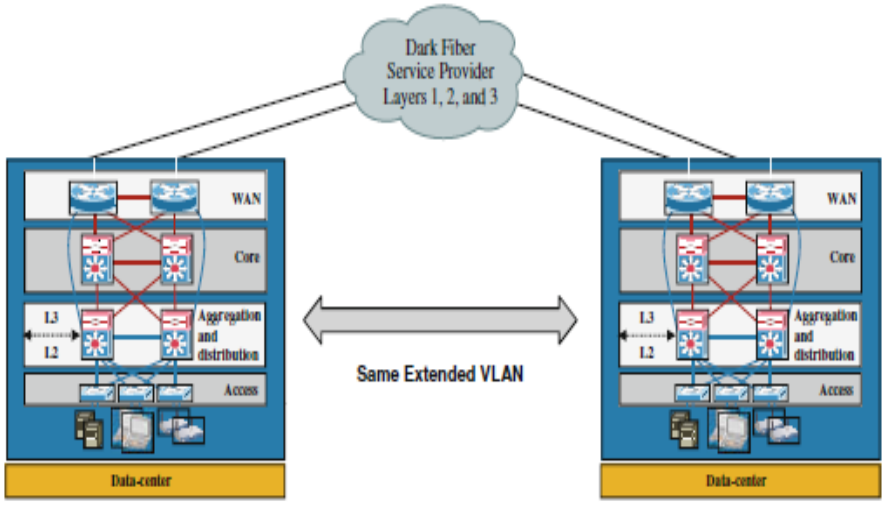


Figure 3.7: Data center interconnect using VLAN

3.1.3 CLOUD IN A BOX

Another component for the design or construction of a hybrid cloud environment is Cloud in a box [16]. This is a self-contained, pre-integrated and pre-packaged platform that can be used to implement cloud centers. The box is made up of multiple blades which perform various activities like computing, switching, storage etc. these blades are interconnected by a backplane and high speed Ethernet connections in a hypervisor environment.

3.1.4 NETWORK SERVICE NODES

Network activities in a hybrid cloud are managed by the Network Service Node [16] which is made up of components that facilitate the communication and security services in a hybrid cloud. Some of the components include; a) Application firewalls that ensures the security of data transport and application workload between data centers in the hybrid cloud environment, b) The server load balancers which ensure that the workload are distributed evenly across data centers, c) WAN accelerators which provide optimization to accelerate the targeted workloads over the WAN. This network service node could be a logical or a physical unit that provide layer 4 network services to enhance cloud services.

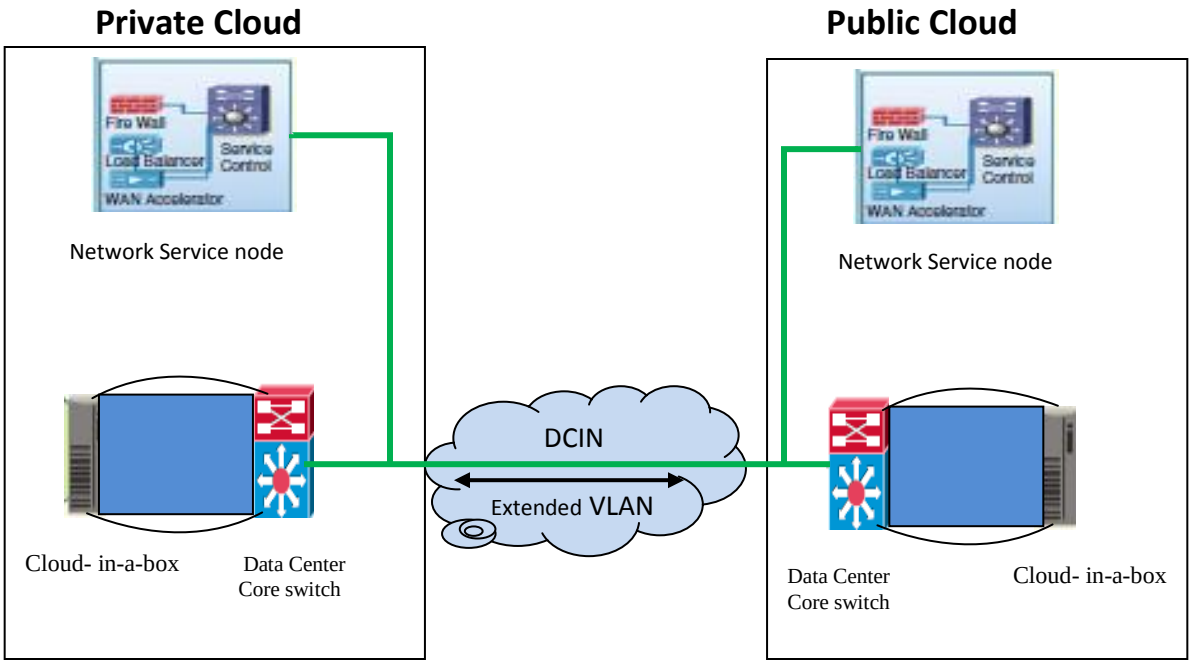


Figure 3.8: The big picture of Hybrid Cloud Architecture

3.2 NETWORK LIMITATIONS IN A HYBRID CLOUD

The major problem with cloud computing especially the public cloud is the security of data in the network. This is why many customers shy away from cloud services especially those dealing with sensitive data like hospitals, banks etc. Kamara and Lauter [22] proposed a secured mechanism, in an attempt to alleviate the security concerns expressed by potential cloud customers. Hybrid cloud leverages the private cloud (data security can be managed by the owner) and the public cloud which uses the public internet. The transfer of data over the network creates challenging security risks. A major concern to the data owner, is the confidentiality of their sensitive data like medical results and financial accounts. A level of assurance is needed to guarantee that the data remains secure when in the data center and in transit. As a solution to the security issue in hybrid cloud, one project "*Cryptographic Cloud Storage"* from the Microsoft Cryptographic Group [22] addresses this issue. The system "Cryptographic Cloud Storage" which can be downloaded (not open source) and installed as an application on the users' machine consists of three modules:

a) *Module 1*: A data processor which processes data that is sent to the cloud

b) *Module 2*: A data verifier that checks whether the data in the cloud has been tampered with

c) *Module 3*: A token generator that generates tokens that enable the cloud service provider to retrieve data, a credential generator that implements an access control policy by issuing credentials to the various parties involve in the system.

Implementation of the system

Consider a case as shown in the diagram below where two users have the potentials of sharing resources in a hybrid cloud environment.

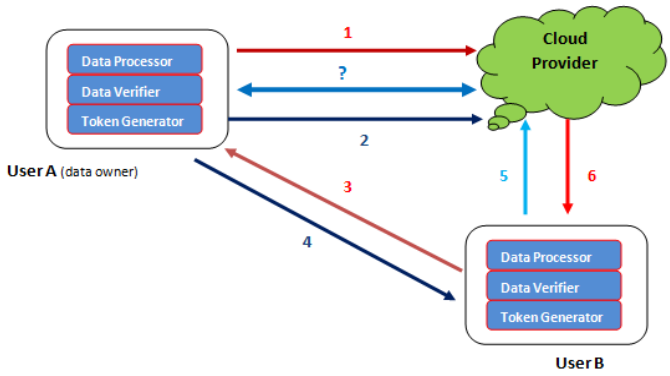


Figure 3.9: Data security and confidentiality process in a cloud environment

In the figure above, **User A** (data owner) wishes to share data to another user **User B** through the Cloud Provider. Upon execution of the software, **User A**'s application generates a cryptographic key (master key) which is stored locally on **User A**'s computer.

Security and confidentiality process in fig 3.8 above

(1) To upload data to the cloud, User A's data processor is invoked which attaches metadata (like size, time and keyword), encrypts and encodes the data and sends to the cloud.

(?) To verify the integrity of data in the cloud at any time, User A's data verifier is invoked which uses the master key to interact with the cloud provider to check the data integrity.

(2) To retrieve the data for modification, the token generator is invoked to create a token, the token is then sent to the cloud provider who uses it to return the encrypted file to User A. User A finally uses its' decryption key to decrypt.

(3) For User A to share data with User B, User B asks for permission to search key word in the cloud. **(4)** User A's token generator is invoked to create a token and the credential generator is invoked to generate a credential for User B. Both the token and the credential are sent to User B who **(5)** then sends the token to the cloud provider. The cloud provider uses the token to retrieve and **(6)** send the appropriate encrypted data to User B who then decrypts the data using his credentials.

It is important to note that this system still has a little bit of loophole as the data modification cannot be done in the cloud since the data in the cloud is encrypted and cannot be accessed. However, the system assures security and confidentiality of data which is a course for concern for many potential cloud users.

CHAPTER FOUR: SIMULATIONS

4.0 INTRODUCTION

The chapter gives the step-by-step procedures, with an evaluation of how pricing (billing) and scalability (virtualization) are achieved in the designed hybrid Cloud Computing environment. The chapter starts by presenting the factors that affect pricing followed by the effects of virtualization in the hybrid cloud environment and ends with the results obtained from the simulations.

4.1 PRICING FACTORS IN A HYBRID CLOUD

Cloud computing be it a Private Cloud, a Public Cloud or a Hybrid Cloud deals with Pay-as-you-go system. This means that customers or users do not bear the cost of the equipment to do their computations, they only pay for services and capacity as they need them, hence the phrase "Pay-as-you-go". The price or bill finally presented to the users depends on the following factors [1]:

4.1.1 ***Storage***: This is the storage space used by users or customers. It is measured in bytes and it is one of the components of the total bill presented to the user. It is calculated thus:

Storage bill (A1) = [Cost per Storage] X [VM's used storage].

4.1.2 ***Memory***: This is the memory size used by the users in their computation. Memory (RAM) is measured in bytes and it contributes to the total bill presented to the user. It is calculated as:

Memory bill (A2) = [Cost per Memory] X [VM's used memory]

4.1.3 ***Compute (CPU)***: This is the processing speed used in the computation. It is expressed in Millions of Instructions Per Second (MIPS) and it is the major contributor to the total bill

borne by the user. It is worth noted that in our daily lives, the processing power is mostly expressed in Hertz. Depending on the architecture (RISC or CISC) and the number of clock cycles it takes to execute an instruction in the said CPU, the corresponding value of MIPS can be determine as: $\boldsymbol{MIPS = \frac{VALUE\ OF\ MHz}{VALUE\ OF\ Clock\ cycles}}$, thus the compute bill is calculated as:

Compute bill (A3) = [Cost per MIPS] X [VM's used MIPS]

4.1.4 ***Bandwidth***: This is the amount of data transferred within the platform during the user's computation. It is measured in bits per seconds (bps) and it also contributes to the total bill presented to the user. It is calculated thus:

Bandwidth bill (A4) = [Cost per Bandwidth] X [VM's used Bandwidth]

$$\boldsymbol{Thus\ the\ total\ bill\ (A5) = \sum_{i=1}^{i=4} Ai}$$

Virtual Machine (VM) is the point of focus in the computation and the time used depends on the provisioning policy (explained in 4.2 below). For example, using a space-shared policy for VM and task, the estimated finished time of a task *p* managed by any VM is given by ;

$$\boldsymbol{eft(p) = est(p) + \frac{rl}{capacity\ X\ Cores(p)}},$$

where;

est(p) is the cloudlet (cloud task) estimated start time which depends on the position of the cloudlet in the execution queue (since space-shared),

rl is the total number of instructions that the cloudlet will need to execute on a processor

cores(p) is the number of cores (PEs) required by the cloudlet,

capacity is the host specification.

In a multi-core system, using space-shared policy, the total capacity of a host having ***np***

processing elements (PEs) is given by;

$$\textbf{Capacity} = \sum_{i=1}^{np} \frac{cap(i)}{np}$$

Where

cap(i) is the processing strength of individual cores (PEs).

On the other hand, using space-shared policy for VM and time-shared policy for task, allows tasks assigned to VMs to context switch during their life cycle. In this case the estimated finish time of a cloudlet *p* managed by a VM is given by

$$\boldsymbol{eft(p) = ct(p) + \frac{rl}{capacity\ X\ Cores(p)}},$$

where

ct(p) is the current simulation time and ***rl, capacity, cores(p)*** are as defined above.

Equally the total capacity for a host in this case is given by:

$$\textbf{Capacity} = \sum_{i=1}^{np} \frac{cap(i)}{(\sum_{j=1}^{cloudlets} Cores(j), np)}$$ since multiple cloudlets (tasks) can simultaneously multi-task within a VM.

4.2 THE EFFECTS OF VIRTUALIZATION ON THE CLOUD COMPUTING ENVIRONMENT.

One of the most important feature of cloud computing is Scalability. This scalability is achieved through virtualization. CloudSim [24], the simulator used for this simulation supports VM provisioning at two levels [18]; first at host level where the overall processing power of each core can be specified and assigned to a VM, and second level which is the VM level where the VM assigns a fixed amount of available processing

power to individual tasks. At each level, CloudSim implements both the space-shared and the time-shared provisioning policies. These provisioning policies include:

a) **Space-shared for both VM and Task**: In this policy, only one VM can run in one core at any instance of time while other VMs will have to queue up. Equally, only one task can run in one core. This implies if a VM uses multi-cores, then many tasks can still run with each in a single core.

b) **Space-shared policy for VM and Time-shared policy for Task**: In this policy, during the VM's life cycle, all tasks are dynamically context switch during their life cycle and only one VM runs on one core at any instance.

c) **Time-shared for VM and Space-shared for Task**: Here each VM uses a time slice on each processing core which then distributes the slices among the tasks on a space share method.

d) **Time-shared for both VM and Task**: In this policy, the processing power is concurrently shared by VMs and the shares of each VM are simultaneously divided among task hence no queuing.

To evaluate the effects of virtualization in this thesis, a time-shared policy is used for both VM and tasks. This is to allow multiple VMs to use a single CPU Core so as to allow parallelism in the execution of multiple tasks. This is tested with executing 80 tasks in our hybrid cloud environment while altering the number of virtual machines.

4.3 SOFTWARE COMPONENTS OF THE SIMULATION

4.3.1 Cloudsim:

Cloudsim [24] is a cloud computing simulation software produced by the GRID LAB of the University of Melbourn in Australia. This software supports the modeling and simulation of

large scale cloud computing infrastructure [14] using a single physical computing node. This software also has the potentials of providing availability of virtualization engine which helps in the creation and management of virtualized services on a data center node. It equally allows the flexibility to switch between Space-shared (i.e assigning specific CPU cores to specific VMs) and Time-shared (i.e dynamically distributing the capacity of a CPU core among VMs) allocation of processing cores to virtualized services. At the time of writing this thesis, Cloudsim 2.1.1 was the current version and it is the version used in this work. Among the threads that were used to generate cloudsim software, two of them [18] are actively managed by Java Virtual Machine (JVM) and they play an active role in this simulation. They include:

4.3.1.1 Datacenter: This class is composed of a set of hosts which are responsible for managing VMs during their life cycle. A host is a component that represents a physical computing node in a cloud with a pre-assigned processing capacity (expressed in MIPS), Memory, Storage, Bandwidth and a scheduling policy for allocating processing cores to VMs. To achieve the goal of evaluating the effect of billing in this thesis, the original Datacenter class in the Cloudsim has been modified by incorporating the pricing policies to implement all the billing factors (MIPS, Memory, Storage and Bandwidth)

4.3.1.2 DatacenterBroker: This class models a broker which is responsible for mediating between users and service providers. The broker acts on behalf of the users to identify suitable cloud service providers through Cloud Information Service (CIS). These two threads above play an active role in the communication among entities in the cloud simulation as seen in the sequence diagram below.

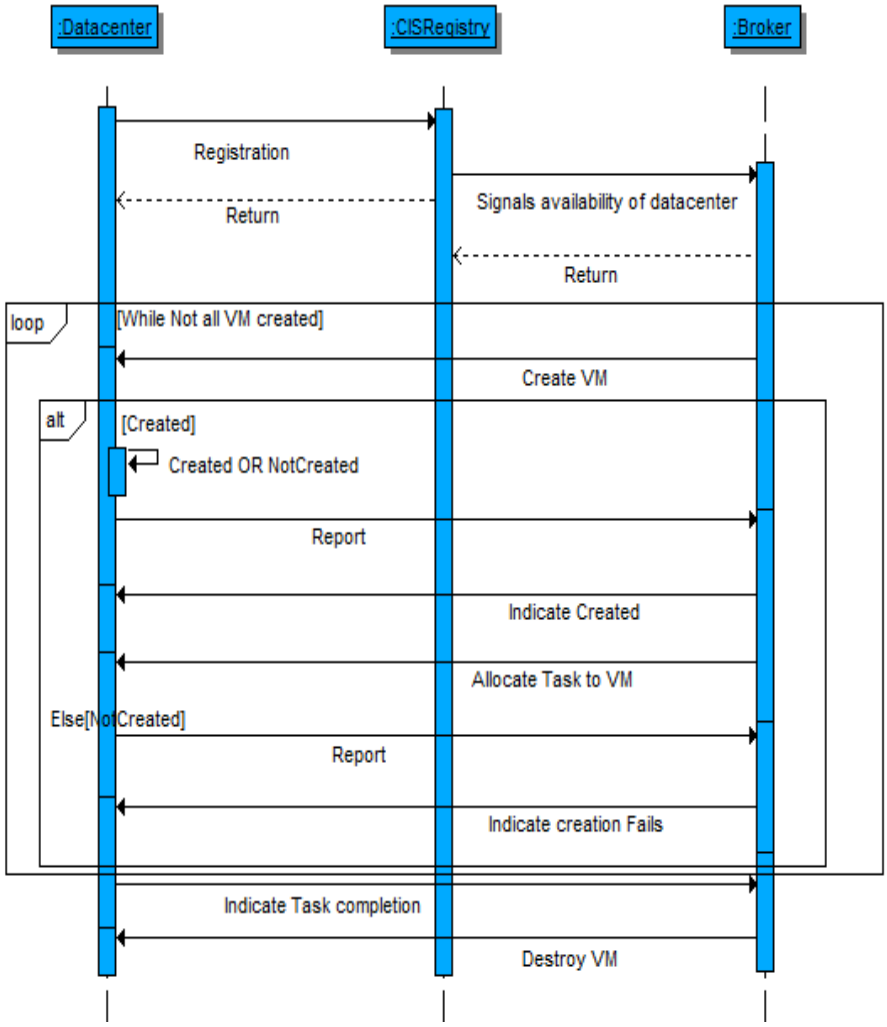


Figure 4.1: Sequence Diagram of the Data Communication Scenario

At the beginning of the simulation, each data entity registers itself with the Cloud Information Service (CIS) Registry. The CIS then provides information for mapping user request to appropriate cloud providers. Brokers, who are acting on behalf of users, consult the CIS service about the list of clouds who offer services matching user's request. The broker then creates VMs, schedule the task to be done and get the bill. When the scheduled task is completed, the datacenter thread informs the broker which then destroys the created VMs to avoid memory leak. These events can be seen in the simulation display in figure 4.2 below.

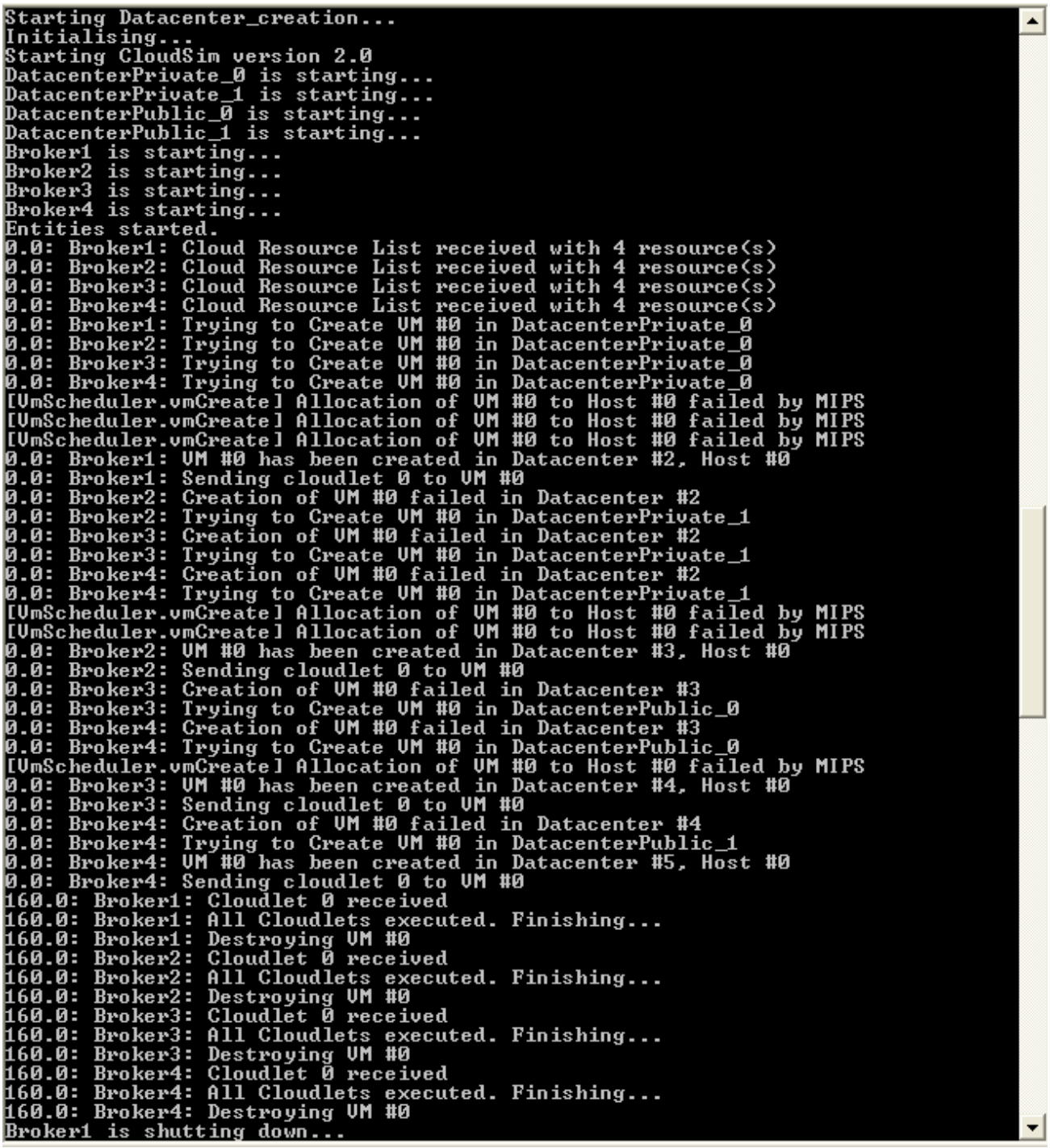


Figure 4.2: Command prompt display of Simulated data flow in the hybrid cloud

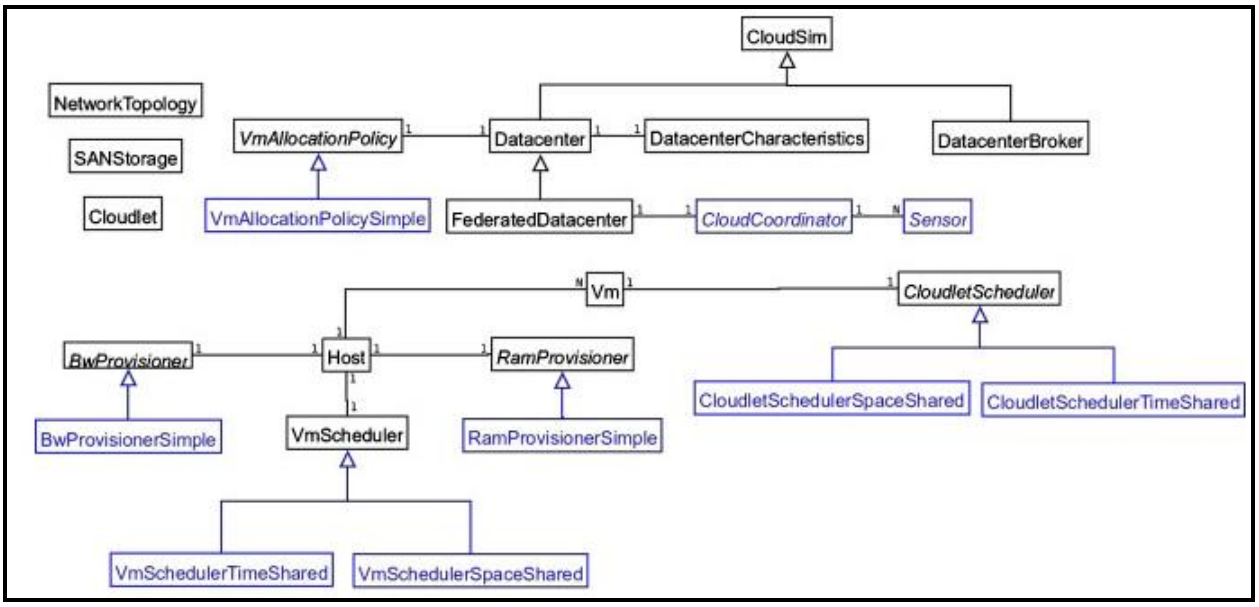


Figure 4.3: Cloudsim class design diagram

4.3.2 Apache Ant

Apache Ant [26] is a java library and command-line tool used to compile codes for this simulation. The version of Apache Ant used for this simulation is Apache Ant 1.8.2 which is the latest version as of the time of writing this thesis.

4.3.3 JDK

JDK is an acronym for Java Development Kit. This JDK is used to develop java applications and run them using Java Run time Environment (JRE). This simulation uses JDK 1.7 because of the condition set by cloudsim 2.1.1 to use JDK 1.6 or higher.

The simulation was also run using Eclipes and Netbeans. In this case there was no need for Apache Ant.

4.4 EXPERIMENTS AND EVALUATION

This section presents the experiments and evaluation conducted in order to evaluate the Pricing mechanism and parallelism strategy in a hybrid cloud. Two sets of simulations were conducted using Cloudsim version 2.1.1 and Apache Ant 1.8.2 (-and equally Eclipse) on an

AMD Sempron machine having the following specifications: a processor speed of 2.10 GHz, L2 cache of 512kb, a RAM size of 2GB and running on Windows 7 Ultimate edition.

4.4.1 Evaluation of Pricing mechanism

The aim of the first set of simulations was to evaluate the factors that affect pricing in a hybrid cloud. The simulated hybrid cloud environment consisted of four datacenters (Two data centers for private cloud and two for public cloud), four brokers and a single user on each data center. Each data center had a single host (for simplicity), which had the following specifications: a single CPU core (1000MIPS), 2GB of RAM, 40GB of storage, 4000bps of bandwidth and running a single task (same task for all). The scheduling policy used here was Space-shared for both VM and time which means only one virtual machine (VM) was used by a single CPU (host) at a time. The VMs on their part ran one task (task size = 4000 MB, task in put size = 300MB, task output size = 300 MB) using the following specifications:

a) 1 CPU core range from 250 MIPS to 1000 MIPS,
b) RAM size range from 512 MB to 2048 MB
c) Storage size range from 10GB to 40GB (using fixed allocation)
d) Bandwidth size range from 1000bps to 4000bps

Below is a command prompt display of one simulation for space shared policy

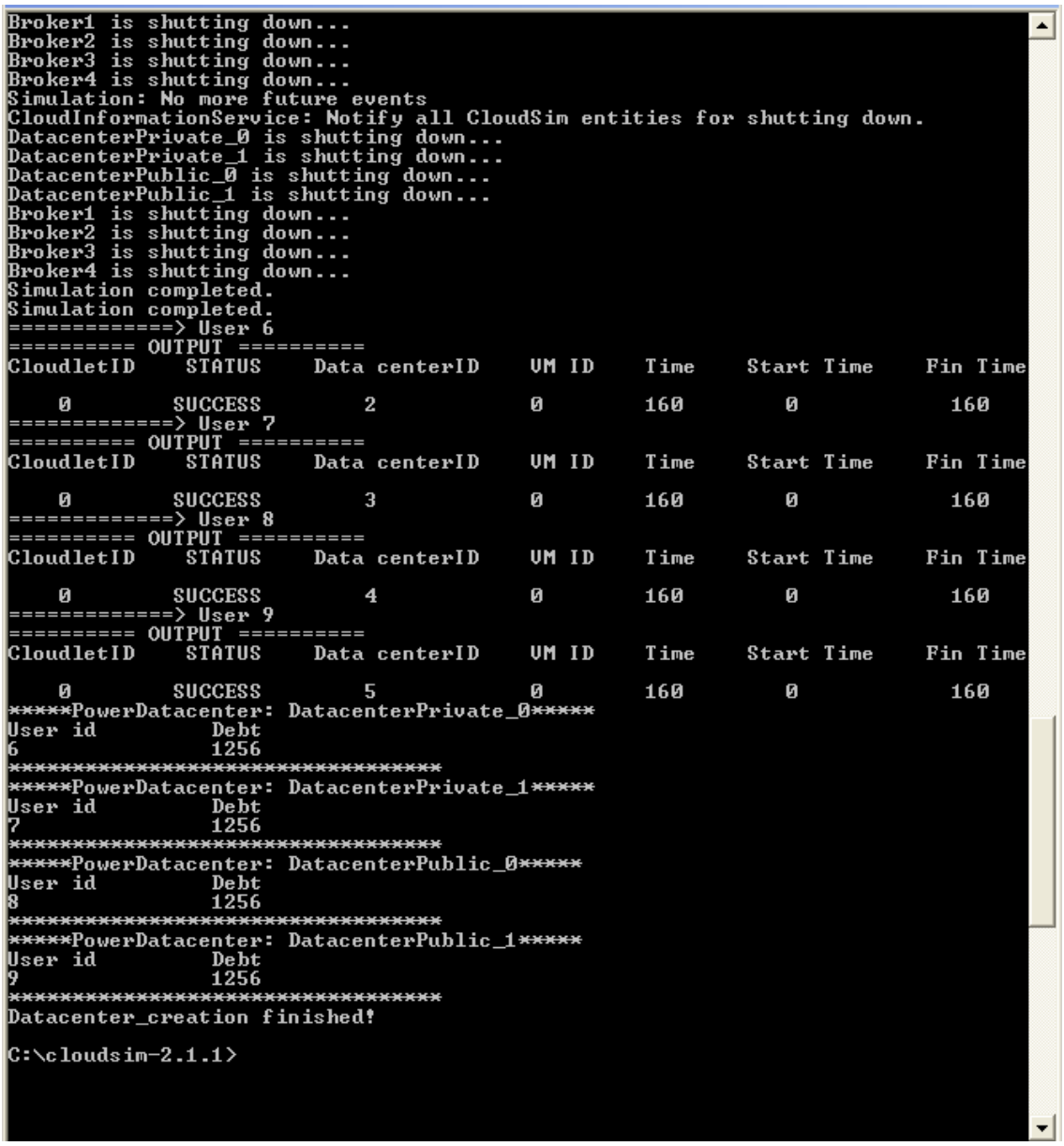
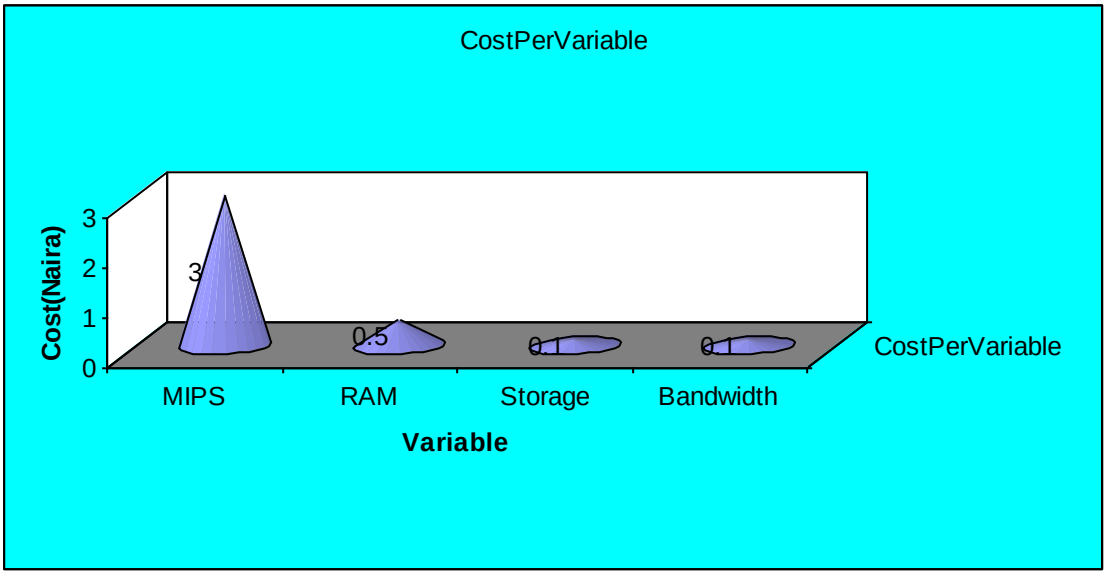


Figure 4.4: Command prompt display of the first simulation.

4.4.1.1 Graphs for pricing mechanism simulation:

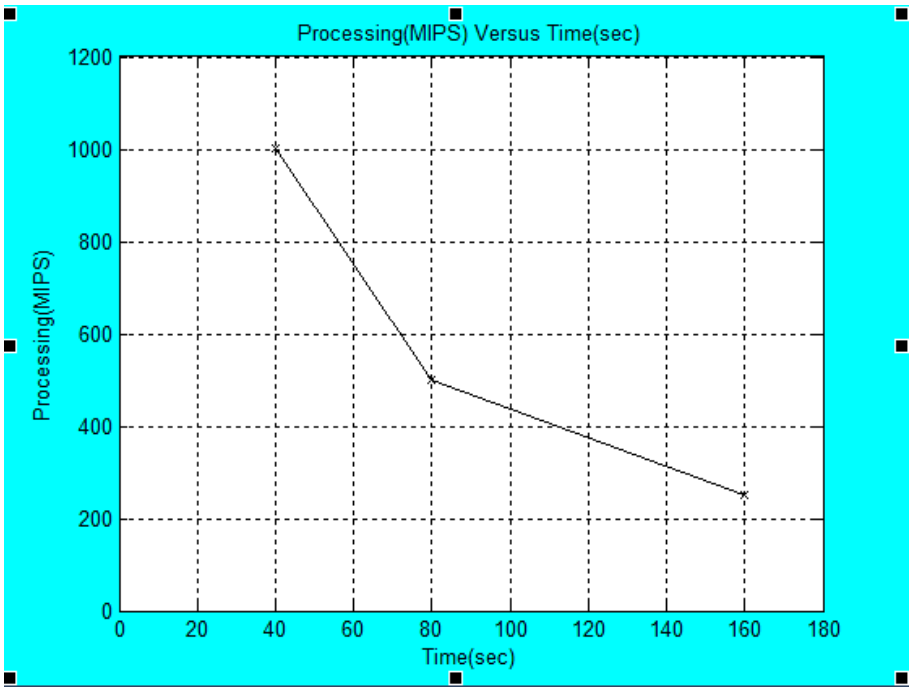
To really evaluate the effect of billing factors in the simulation, each factor was varied and others kept constant in turns. The factors were each given a fixed cost per unit which is then used to finally determine the debt based on the quantity of units used in the computation.

Latency or time delay is not really considered as a factor in the simulation because it depends on the used value of MIPS; MIPS value is inversely proportional to the time used and directly proportional to the debt incurred. The evaluation can be appreciated from the graphs below



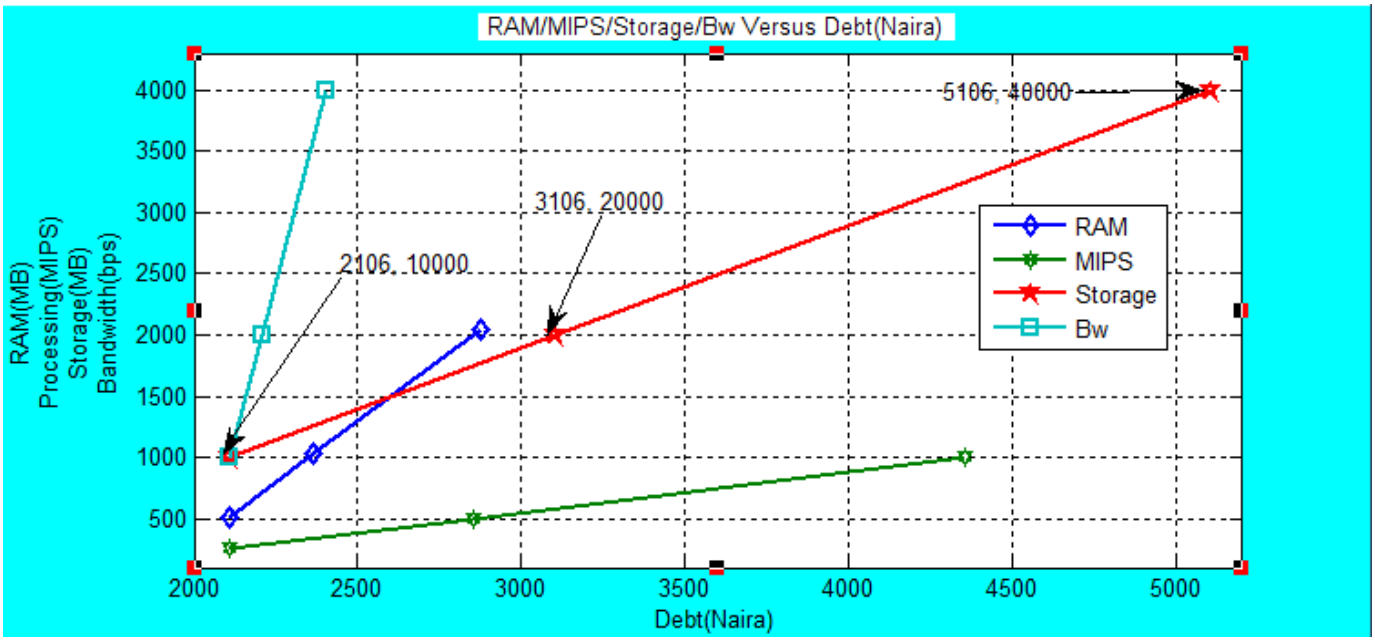
Graph 4.1: Cost Per Unit of Resources/Month used in the simulation

a) **Cost Per Resource/Month**: The factors were initially given a fixed cost per unit ranging from compute intensive to data intensive factors with the following assignment: Cost Per MIPS = 3 Naira, Cost Per RAM = 0.5 Naira, Cost Per Storage = 0.1 Naira and Cost Per Bandwidth = 0.1Naira



Graph 4.2: MIPS Vs Time

b) **MIPS Vs TIME**: MIPS is inversely proportional to the time delay (latency). Therefore, increasing the value of MIPS will reduce the time used for the computation which will equally increase the compute bill and verse versa.



Graph 4.3: RAM/MIPS/Storage/Bw Vs Debt

c) **CPU Vs Debt**: Here the following MIPS values were used: 250, 500, 1000, while the other factors were kept constant with the following values: RAM = 512MB, Storage = 10000 MB, Bandwidth = 1000bps

d) **RAM Vs Debt**: Here all other factors were kept constant with the following values MIPS = 250 Hz, Storage = 10000 MB, Bandwidth = 1000 bps while the following values for RAM were used: 512, 1024, 2048.

e) **Storage Vs Debt**: Here values used for Storage were 10000, 20000,40000 while other factors were kept constant with the following values: RAM = 512 MB, MIPS = 250 Hz, Bandwidth = 1000 bps

f) **Bandwidth Vs Debt**: To equally evaluate Bandwidth effect in debt calculation, all other factors were kept constant with the following values: MIPS = 250Hz, RAM = 512MB, Storage = 10000MB, while Bandwidth was simulated with the following values: 1000, 2000, 4000

Graph 4.3 presents the different values of processing power against debt, different values of RAM against debt, different values of Storage against debt and different values of Bandwidth against debt. It is observed from the graph that the values of RAM, Storage, MIPS or Bandwidth assigned to a virtual machine cannot be more than the corresponding values assigned to the host. The results indicate a linear relationship with different slopes between the pricing factors and the debt incurred. It could also be observed from the graph that Per unit MIPS was the main contributor of the bill whereas Per unit Bandwidth was the least contributor since there was no data transfer during the simulation

4.4.2 Evaluation of Hybrid Cloud parallelism Strategy

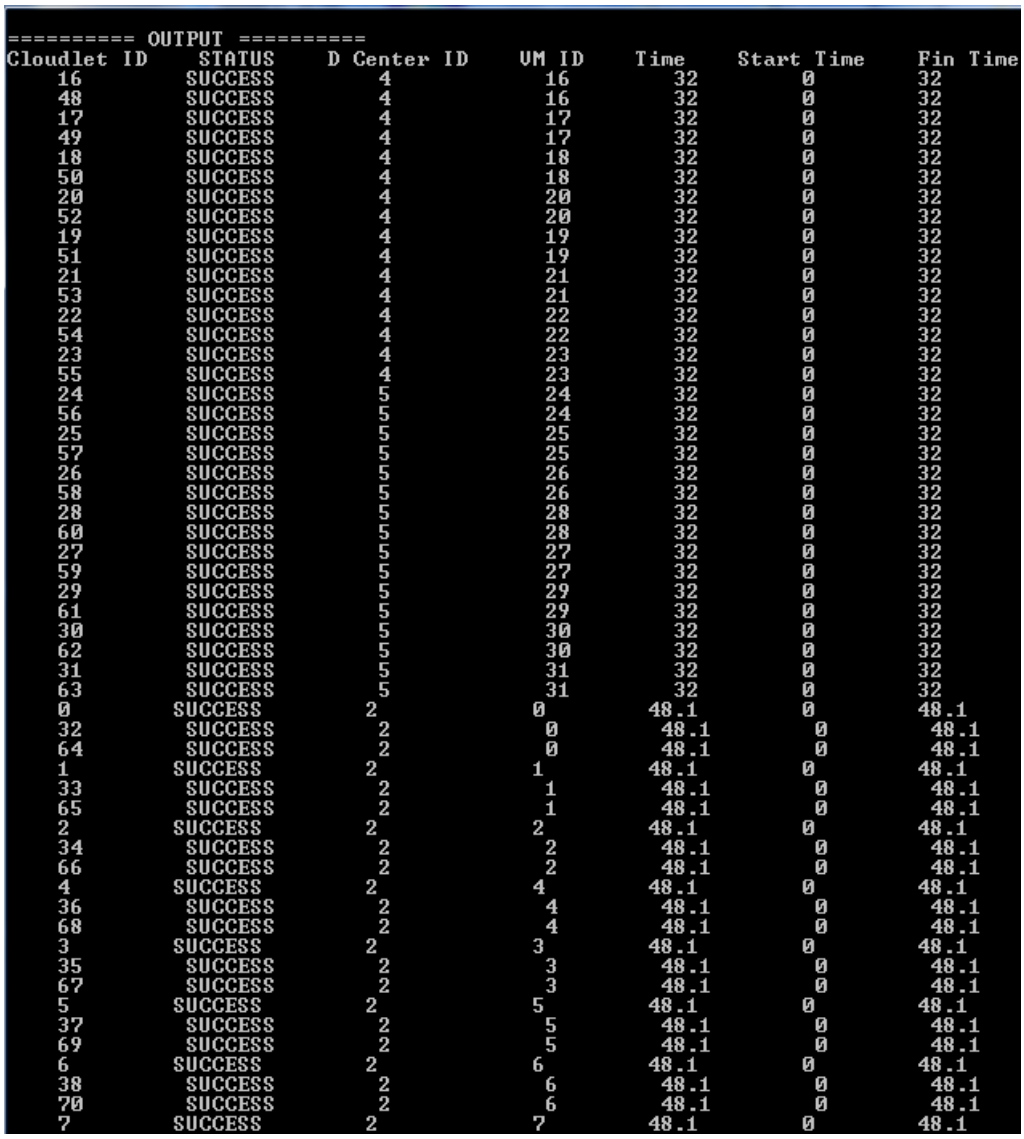
The second set of simulations was to evaluate the effect of parallelism in task execution in a hybrid cloud. The simulation scenario models a network of private and public cloud each modeled to have two data centers and the same specifications as in 4.4.1 above but this time around using time-shared policy for VMs and tasks with a range of 10 to 40 VMs and 80 tasks involved in the simulation. The data from the simulation can be seen in the table below.

Table 4.1: Effect of Virtualization in a Hybrid cloud environment

No of VM per Host	No of Tasks	Datacenter ID	Datacenter concerned	No of Users(Hosts)	Time(sec)	Policy
40	32	4	Public	1	32	Time-Shared
		5		1	32	
	48	2	Private	1	48.1	
		3		1	48.1	

From the table above, it is noticed that the execution time is highly favoured with increase in the number of Virtual Machines as compared to the time of a single VM per host. In the simulation with 40 VMs, the first group of 32 tasks was able to complete their execution in 32 seconds while the other 48 tasks completed in 48.1 seconds. This disparity in the time of completion is because of time shared policy which allows the dynamic distribution of VM in the host. Thus in the beginning of the execution, the hosts Public data centers were not overloaded as they had only 32 tasks of the 80 tasks in the simulation while in the Private data centers, there was an increase in the number of tasks. This experiment showed that the use of a hybrid cloud computing environments could boost the productivity of a company as companies can expand

their capacity by leasing the resources from one of their clouds (private or public) to use in the other as seen in the figure below where in the simulation of 80 tasks, 32 were handled by the Public cloud while 48 were handled by the Private cloud simultaneously.



```
========== OUTPUT ==========
Cloudlet ID    STATUS    D Center ID    VM ID    Time    Start Time    Fin Time
    16         SUCCESS         4           16       32         0          32
    48         SUCCESS         4           16       32         0          32
    17         SUCCESS         4           17       32         0          32
    49         SUCCESS         4           17       32         0          32
    18         SUCCESS         4           18       32         0          32
    50         SUCCESS         4           18       32         0          32
    20         SUCCESS         4           20       32         0          32
    52         SUCCESS         4           20       32         0          32
    19         SUCCESS         4           19       32         0          32
    51         SUCCESS         4           19       32         0          32
    21         SUCCESS         4           21       32         0          32
    53         SUCCESS         4           21       32         0          32
    22         SUCCESS         4           22       32         0          32
    54         SUCCESS         4           22       32         0          32
    23         SUCCESS         4           23       32         0          32
    55         SUCCESS         4           23       32         0          32
    24         SUCCESS         5           24       32         0          32
    56         SUCCESS         5           24       32         0          32
    25         SUCCESS         5           25       32         0          32
    57         SUCCESS         5           25       32         0          32
    26         SUCCESS         5           26       32         0          32
    58         SUCCESS         5           26       32         0          32
    28         SUCCESS         5           28       32         0          32
    60         SUCCESS         5           28       32         0          32
    27         SUCCESS         5           27       32         0          32
    59         SUCCESS         5           27       32         0          32
    29         SUCCESS         5           29       32         0          32
    61         SUCCESS         5           29       32         0          32
    30         SUCCESS         5           30       32         0          32
    62         SUCCESS         5           30       32         0          32
    31         SUCCESS         5           31       32         0          32
    63         SUCCESS         5           31       32         0          32
    0          SUCCESS         2           0        48.1       0          48.1
    32          SUCCESS         2           0        48.1       0          48.1
    64          SUCCESS         2           0        48.1       0          48.1
    1          SUCCESS         2           1        48.1       0          48.1
    33          SUCCESS         2           1        48.1       0          48.1
    65          SUCCESS         2           1        48.1       0          48.1
    2          SUCCESS         2           2        48.1       0          48.1
    34          SUCCESS         2           2        48.1       0          48.1
    66          SUCCESS         2           2        48.1       0          48.1
    4          SUCCESS         2           4        48.1       0          48.1
    36          SUCCESS         2           4        48.1       0          48.1
    68          SUCCESS         2           4        48.1       0          48.1
    3          SUCCESS         2           3        48.1       0          48.1
    35          SUCCESS         2           3        48.1       0          48.1
    67          SUCCESS         2           3        48.1       0          48.1
    5          SUCCESS         2           5        48.1       0          48.1
    37          SUCCESS         2           5        48.1       0          48.1
    69          SUCCESS         2           5        48.1       0          48.1
    6          SUCCESS         2           6        48.1       0          48.1
    38          SUCCESS         2           6        48.1       0          48.1
    70          SUCCESS         2           6        48.1       0          48.1
    7          SUCCESS         2           7        48.1       0          48.1
```

Figure 4.5: Command prompt display of the simulation of 40 Virtual machines and 80 tasks

CHAPTER FIVE: CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The recent works of Rodrigo et al in [14] and [18] on cloudsim did not incorporate pricing policies. To fill this gab, we have extended the cloudsim tool kit by implementing a pricing mechanism that effectively manages the bills in the cloud environment. The efficiency of this new implementation has been tested in this thesis to evaluate the effect of the pricing factors in the hybrid cloud. This work will therefore serve as a tool for future research on cloud management thereby improving on the number of prospective cloud users.

5.2 FUTURE WORKS

In our design of the network architecture for a hybrid cloud, we concentrated on the network architecture of the immediate vicinity of the cloud provider, paying little attention on what happens en route when the packets are transmitted between the provider and the users. This security issue is one of the main setbacks of cloud computing. This is because most customers (companies) of cloud computing deal with sensitive and confidential data that if leaked to a third party, the company may suffer a severe loss. So as future work, we plan to work on the best approach to improve on the confidentiality and security of data when hosted in the cloud.

REFERENCES

[1] Borko Furht. Cloud Computing Fundamentals. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[2] Victor Delgado. Exploring the limits of Cloud Computing. *MSc. Thesis*, Stockholm 2010

[3] Hai Jin, Shadi Ibrahim, Tim Bell, Wei Gao, Dachuam Huang, Song Wu. Cloud Types and Services. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[4] The Art of Service: Cloud Computing Specialist Certification Kit; Virtualization. http://www.theartofservice.com/. October 2011

[5] Jinzy Zhu. Cloud Computing Technologies and Applications. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[6] Cisco Data Center Infrastructure 2.5 Design Guide. http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf. October 2011

[7] Anthony M. Middleton. Data-Intensive Technologies for Cloud Computing. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[8] Cisco Data Center Interconnect Design and Implementation Guide. *www.**cisco**.com/en/.../**data_center_interconnect_design_guide**.pdf*. September 2011

[9] David Villegas, Ivan Rodero, Liana Fong, Norman Bobroff, Yandbin Liu, Manish P. and Sadjack S. The Role of Grid Computing Technologies in the Cloud Computing. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[10] Buyya R, Rajiv R, Rodrigo N. Calheiros. Modeling and Simulation of Scalable Cloud Computing Environments and the Cloudsim Toolkit: Challenges and Opportunities, *Proceedings of 7th High Performance Computing and Simulation Conference (HPCS 09), IEEE Computer Society*, June 2009

[11] Cisco Managed Versus Unmanaged Switches.

http://www.cisco.com/go/switching. September 2011

[12] Raffaele Montella, Ian Foster. Using Hybrid Grid/Cloud Computing Technologies for Environmental Elastic STORAGE, Processing and Provisioning. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[13] Managing better with Managed Switches. http://en-wikipedia.org/wiki/managed-switches.pdf. September 2011

[14] Rodrigo N. Calheiros, Rajiv R., Anton B. Cesar A. Buyya R. CloudSim: a toolkit for Modeling and Simulation of Cloud Computing environment and Evalation of resource provisioning algorithms; softw. Pract. Exper. 2011; **41**:23-50. *Wiley Online Library (wileyonlinelibrary.com).* August 24th, 2010

[15] Krishma Kant. Data Center evolution. *ELSEVIER B.V (*www.elsevier.com/locate/comet*)*, 2009

[16] Gen Lin, Mac Devine. The Role of Networks in Cloud Computing. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[17] Hohamad Al-Fares, Alexander L, Vahdat A. A Scalable Commodity Data Network Architecture. *SIGCOMM'08*, August 17-22, 2008

[18] Rodrigo N. Calheiros, Cesar A., Buyya R. CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructure and Services. *Wiley Press*, USA, 2010

[19] Buyya R. Rajiv R., Rodrigo N. Calheiros. InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling Application Services. *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010), Springer.* Germany, 21-23 May 2010

[20] Mladen A. Vouk, Eric Sills, Patrick Dreher. Integration of High Performance Computing into Cloud Computing Services. *Hand Book of Cloud Computing, Springer Science + Business Media*, LLC 2010

[21] k. Mukherjee, S. Sahoo. Development of Mathematical Model for Market-Oriented Cloud Computing. *International Journal of Computer Applications (0975-8887), Volume 9-No 11*, November 2010.

[22] Seny Kamara, Kristin Lauter. Cryptographic Cloud Storage. *Proceedings of Workshop on Real Life Cryptography Protocols and Standardization 2010*, January 2010

[23] Dai Yuefa, Wu Bo, Gu Y, Zhang Q, Tang Chaging. Data Security Model for Cloud Computing. *Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009)*, November 21-22, 2009

[24] Wikipedia . Cloudsim 2.1.1 http://www.cloudbus.org/cloudsim/. September 2011

[25] R. Buyya, C. S. Yeg, S. Venugopal. Market-Oriented Cloud Computing: Vision, hype, and reality for delivering IT Services as Computing Utilities. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, 2008

[26] Wikipedia . Apache Ant. http://ant.apache.org/. September 2011

[27] Wikipedia. Differences between Grid and Cloud http://www.ibm.com/developerworks/web/library/wa-cloudgrid/. September 2011

[28] Wikipedia, Cisco Price List http://www.1st-computer-networks.co.uk/cisco-complet.php. October 2011

[29] C.E Leiserson. Universal Networks for Hardware-efficient supercomputing. *IEEE Transactions on computers*. 1985

APPENDIX

Extended Cloud simulator code and simulation code for pricing mechanism

1. *Extended CloudSim code for pricing mechanism*

```
package hybridCloud.thesis;

/* Importation from java library*/
import java.util.List;

/* Importation from cloudsim library */
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSimTags;
import org.cloudbus.cloudsim.core.SimEvent;

/* Class for the extended data center*/
public class DataCentreModified extends Datacenter{

/** constructor for new extended data center**/
	public DataCentreModified(String arg0, DatacenterCharacteristics arg1,
			VmAllocationPolicy arg2, List<Storage> arg3, double arg4)
			throws Exception {
		super(arg0, arg1, arg2, arg3, arg4);
		// TODO Auto-generated constructor stub
	}

	/** Method to process virtual machine migration**/
	protected void processVmMigrate(SimEvent ev, boolean ack) {

	}

	/** Method to process virtual machine creation,
	    gets its data and process the billing **/
	protected void processVmCreate(SimEvent ev, boolean ack) {
	Vm vm = (Vm) ev.getData();

	    boolean result = getVmAllocationPolicy().allocateHostForVm(vm);

	    if (ack) {
	       int[] data = new int[3];
	     data[0] = getId();
	       data[1] = vm.getId();

	     if (result) {
	          data[2] = CloudSimTags.TRUE;
	     } else {
	          data[2] = CloudSimTags.FALSE;
	     }
	          sendNow(vm.getUserId(), CloudSimTags.VM_CREATE_ACK, data);
	    }

	    if (result) {
			double amount = 0.0;
			if (getDebts().containsKey(vm.getUserId())) {
				amount = getDebts().get(vm.getUserId());
```

```
                    }
                    amount += getCharacteristics().getCostPerMem() * vm.getRam();
                    amount += getCharacteristics().getCostPerStorage() * vm.getSize();
                    amount += getCharacteristics().getCostPerMi() * vm.getMips();
                    amount += getCharacteristics().getCostPerBw() * vm.getBw();

                    getDebts().put(vm.getUserId(), amount);

                    getVmList().add(vm);

            }

    }


}
```

2. *Simulation code for the pricing mechanism*

```
package hybridCloud.thesis;

// Importation from java library
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

// Importation from cloud simulator library
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicy;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerSpaceShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.core.CloudSimTags;
import org.cloudbus.cloudsim.core.SimEvent;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;


/** Class to create data centers in a hybrid cloud
 */
public class Datacenter_creation {

       /** The creation of tasks lists. */
       private static List<Cloudlet> cloudletList1;
       private static List<Cloudlet> cloudletList2;
       private static List<Cloudlet> cloudletList3;
       private static List<Cloudlet> cloudletList4;
```

```
	/** The creation of virtual machine lists. */
	private static List<Vm> vmlist1;
	private static List<Vm> vmlist2;
	private static List<Vm> vmlist3;
	private static List<Vm> vmlist4;


	// The main method to run the simulation
	public static void main(String[] args) {

		Log.printLine("Starting Datacenter_creation...");

		try {
			// Initialization of the CloudSim package.
			int num_user = 1;   // number of cloud users
			Calendar calendar = Calendar.getInstance();
			boolean trace_flag = false;  // mean trace events

			// Initialization of the CloudSim library
			CloudSim.init(num_user, calendar, trace_flag);

			// Creation of Data centers (resource providers in CloudSim.)
			DataCentreModified datacenter0 =
createDatacenter("DatacenterPrivate_0");
			DataCentreModified datacenter1 =
createDatacenter("DatacenterPrivate_1");

			DataCentreModified datacenter2 =
createDatacenter("DatacenterPublic_0");
			DataCentreModified datacenter3 =
createDatacenter("DatacenterPublic_1");

			//Creation of Brokers (act on behalf of users)
			DatacenterBroker broker1 = createBroker(1);
			int brokerId1 = broker1.getId();

			DatacenterBroker broker2 = createBroker(2);
			int brokerId2 = broker2.getId();

			DatacenterBroker broker3 = createBroker(3);
			int brokerId3 = broker3.getId();

			DatacenterBroker broker4 = createBroker(4);
			int brokerId4 = broker4.getId();


			//Creation of virtual machines, one for each broker
			vmlist1 = new ArrayList<Vm>();
			vmlist2 = new ArrayList<Vm>();
			vmlist3 = new ArrayList<Vm>();
			vmlist4 = new ArrayList<Vm>();


			// Virtual machine specification
			int vmid = 0; // virtual machine ID
			int mips = 250; // MIPS assigned value to the VM
			long size = 10000; //storage(MB) size of the VM
			int ram = 512; //Memory(MB) assigned value to the VM
			long bw = 1000; // Bandwidth assigned value to the VM
			int pesNumber = 1; //number of CPUs in the VM
			String vmm = "Xen"; //VM Monitor name
```

```
                    //Allocation of memory to the VMs
                    Vm vm1 = new Vm(vmid, brokerId1, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());
                    Vm vm2 = new Vm(vmid, brokerId2, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());
                    Vm vm3 = new Vm(vmid, brokerId3, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());
                    Vm vm4 = new Vm(vmid, brokerId4, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());

                    //add the VMs to the vmlists
                    vmlist1.add(vm1);
                    vmlist2.add(vm2);
                    vmlist3.add(vm3);
                    vmlist4.add(vm4);

                    //submit vm list to the broker
                    broker1.submitVmList(vmlist1);
                    broker2.submitVmList(vmlist2);
                    broker3.submitVmList(vmlist3);
                    broker4.submitVmList(vmlist4);

                    //Creation of Cloudlets (tasks)
                    cloudletList1 = new ArrayList<Cloudlet>();
                    cloudletList2 = new ArrayList<Cloudlet>();
                    cloudletList3 = new ArrayList<Cloudlet>();
                    cloudletList4 = new ArrayList<Cloudlet>();

                    //cloudlets (task) specification
                    int id = 0;
                    long length = 40000; //task size(MB)
                    long fileSize = 300; //task input size (MB)
                    long outputSize = 300; //task output size (MB)
                    UtilizationModel utilizationModel = new UtilizationModelFull();

                    // Allocation of memory to cloudlets (tasks)
                    Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet1.setUserId(brokerId1);
                    Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet2.setUserId(brokerId2);
                    Cloudlet cloudlet3 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet3.setUserId(brokerId3);
                    Cloudlet cloudlet4 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet4.setUserId(brokerId4);

                    //add the cloudlets to the lists
                    cloudletList1.add(cloudlet1);
                    cloudletList2.add(cloudlet2);
                    cloudletList3.add(cloudlet3);
                    cloudletList4.add(cloudlet4);

                    //submit cloudlet list to the brokers
                    broker1.submitCloudletList(cloudletList1);
                    broker2.submitCloudletList(cloudletList2);
                    broker3.submitCloudletList(cloudletList3);
                    broker4.submitCloudletList(cloudletList4);

                    // Starting the simulation
                    CloudSim.startSimulation();
```

```
                // Printing the results when simulation is over
                List<Cloudlet> newList1 = broker1.getCloudletReceivedList();
                List<Cloudlet> newList2 = broker2.getCloudletReceivedList();
                List<Cloudlet> newList3 = broker3.getCloudletReceivedList();
                List<Cloudlet> newList4 = broker4.getCloudletReceivedList();

                CloudSim.stopSimulation();

                Log.print("==============> User "+brokerId1+"    ");
                printCloudletList(newList1);

                Log.print("==============> User "+brokerId2+"    ");
                printCloudletList(newList2);

                Log.print("==============> User "+brokerId3+"    ");
                printCloudletList(newList3);

                Log.print("==============> User "+brokerId4+"    ");
                printCloudletList(newList4);


                //Printing the debt of each user to corresponding data center
                datacenter0.printDebts();
                datacenter1.printDebts();
                datacenter2.printDebts();
                datacenter3.printDebts();
                Log.printLine("Datacenter_creation finished!");
            }
            catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
            }
        }
        // Method for the creation of data center
        private static DataCentreModified createDatacenter(String name){

            // creation of a list to store our machine
            List<Host> hostList = new ArrayList<Host>();

            // assignment of mips to host machines
            List<Pe> peList = new ArrayList<Pe>();

            int mips=1000;

            //Creating Processing Elements and adding into a list.
            peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating

            //Creation of Host with its specifications
            int hostId=0;   // host ID
            int ram = 2048; //host memory (MB)
            long storage = 40000; //host storage
            int bw = 4000; //host bandwidth

            // adding host in the list using space-shared policy
            hostList.add(
                new Host(
                    hostId,
                    new RamProvisionerSimple(ram),
                    new BwProvisionerSimple(bw),
                    storage,
                    peList,
```

```
                        new VmSchedulerSpaceShared(peList)
                    )
            );

            //   setting properties for DatacenterCharacteristics
            String arch = "x86";      // system architecture
            String os = "Windows 7"; // operating system
            String vmm = "Xen";  // virtual machine monitor
            double time_zone = 1.0;  // time zone this resource located
            double costPerMi = 3.0; // the cost of using processing in this resource
            double costPerMem = 0.5;   // the cost of using memory in this resource
            double costPerStorage = 0.1;// the cost of using storage in this resource
            double costPerBw = 0.1;    // the cost of using bw in this resource
            LinkedList<Storage> storageList = new LinkedList<Storage>();

            DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
               arch, os, vmm, hostList, time_zone, costPerMi, costPerMem,
costPerStorage, costPerBw);


            // creation of Datacenter object.
            DataCentreModified datacenter = null;
            try {
                  datacenter = new DataCentreModified(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }

            return datacenter;
      }

      // Method to create Broker
      private static DatacenterBroker createBroker(int id){

            DatacenterBroker broker = null;
            try {
                  broker = new DatacenterBroker("Broker"+id);
            } catch (Exception e) {
                  e.printStackTrace();
                  return null;
            }
            return broker;
      }


       //Method to print the cloudlet objects
      private static void printCloudletList(List<Cloudlet> list) {
            int size = list.size();
            Cloudlet cloudlet;

            String indent = "    ";
            Log.printLine();
            Log.printLine("========== SIMULATION OUTPUT ==========");
            Log.printLine("CloudletID" + indent + "STATUS" + indent +
                        "Data centerID" + indent + "VM ID" + indent + "Time" +
indent + "Start Time" + indent + "Fin Time");

            DecimalFormat dft = new DecimalFormat("###.##");
            for (int i = 0; i < size; i++) {
                  cloudlet = list.get(i);
                  Log.print(indent + cloudlet.getCloudletId() + indent + indent);
```

```
                    if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
                        Log.print("SUCCESS");

                        Log.printLine( indent + indent + cloudlet.getResourceId() +
indent + indent + indent + cloudlet.getVmId() +
                                        indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent + indent +
dft.format(cloudlet.getExecStartTime())+
                                        indent + indent + indent +
dft.format(cloudlet.getFinishTime()));
                    }
            }

        }
}
```