# Stratus: Building and Evaluating a Private Cloud for a Real-World Financial Application

by

Deepak Bajpai

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg

Thesis advisor

**Ruppa K. Thulasiram**

Author

**Deepak Bajpai**

**Stratus: Building and Evaluating a Private Cloud for a Real-World Financial Application**

# Abstract

Cloud computing technology has been emerging and spreading at a great pace due to its service oriented architecture, elasticity, cost-effectiveness, etc. Many organizations are using Infrastructure-as-a-Service (IaaS) public Clouds for data migration away from traditional IT Infrastructure but there are a few fields such as finance, hospitals, military and others that are reluctant to use public Clouds due to perceived security vulnerability. Enterprises in such fields feel more vulnerable to security breaches and feel secure using in-house IT infrastructure. The introduction of private Clouds is a solution for these businesses.

Private Clouds have been substituted for the traditional IT Infrastructure due to its flexible "pay-as-you-go" model within an organization by departments and enhanced privacy relative to public Clouds in the form of administration control and supervision.

Goal of my thesis is to build and evaluate a private Cloud that can provide virtual machines (VMs) as a service and applications as a service. To achieve this goal, in my thesis, I have built and evaluated a service oriented IaaS model of private Cloud. I have used off-the-shelf servers and open-source software for this purpose. I have

proposed a new replication strategy using an Openstack component called Cinder. My experiments show that efficient VM failure recovery on the basis of "preparation delay" time can be achieved using my strategy. I have studied a real-world application of option pricing from the finance market and have used that application for the purpose of testing my private Cloud for compute workload and accuracy of the pricing results. Later, I have compared performance between Cloud VMs and standalone servers. The performance of Cloud VM is found to be better to standalone servers as long as the number of virtual CPUs (vCPUs) are limited to single node.

Stratus clouds are groups of small clouds that collectively give a spectacular sight in the sky. The private Cloud I have built uses multiple small modules to achieve the stated goal and hence I named my private Cloud "Stratus". This Stratus private Cloud is now ready for deploying applications, and for providing VMs on-demand.

# Acknowledgments

There's no one I'm indebted to more than my advisor Dr. Ruppa Thulasiram, for guiding me, encouraging me, extending his intelligence, patience to my work's completion, and for being there always at the time of crisis where my composure was weak.

Many others have acted generously towards me and my work. I would like to thank Dr. Peter Graham, who gave his time and ink to correct my technical writing and presentation and I am also thankful to Dr. Ian Jeffrey for his keen interest and thorough and critical review of my proposed work.

I'm extremely thankful to my parents Sushil Kumar Bajpai and Kusum Bajpai for having an utmost belief in my commitment and efforts. They have encouraged me since the beginning of my education to put my knowledge towards meaningful work. It would have been impossible without them to accomplish this work. Many thanks to my sister Deepti Tiwari for being strict but at the same time being my best friend to help me to keep going.

I would like to thank Abhilasha for pushing me to pursue this work and motivating me to make it happen. My special thanks to my lab-mates Gobind and Jose for their help in my stressful times and for all cheerful memories that we celebrated in our lab. Thanks to all the students of computer science department and technical staff for their help and encouragement in my work. I would also like to thank Arsenal Football Club which was responsible for my 70% of happiness over the past year.

Last but not the least, I would like to thank that extreme power of God which infused me with self motivation to complete this work.

*This thesis is dedicated to my late grandparents. Although, I have never seen them, but I can feel their blessing with me always.*

# Contents

# List of Figures

# List of Tables

# Glossary

**Cloud computing:** Practice to store, access and compute information from anywhere on physical server, storage and network.

**Hypervisor:** Software to run virtual machines.

**Public Cloud:** Service oriented architecture where the Cloud provider provides resources in the form of services to customers.

**Private Cloud:** On-premise Cloud solution where virtual machines are hosted in an organization's infrastructure for privacy and security concerns.

**Virtual machine (VM):** Virtual machines (VMs) are the virtual desktop or server created on physical resources using hypervisor.

**Virtualisation:** It is a process of creating a virtual (not real) resources for system components like server, storage, network, etc.,.

**VM consolidation:** Optimal resource utilization to create maximum number of virtual machines in a physical system.

**VM migration:** Moving a VM from one physical (real) machine to another.

# Chapter 1

# Introduction

With the exponential growth of Information Technology (IT) Infrastructure and increasing scalability cost of IT from the small scale to high end enterprise sectors, a technical strategy to effectively reduce the cost associated with IT infrastructure has become essential. As IT expanded from mainframe computing to Grid and now Cloud computing, this resulted in benefits such as easy operability, less migration cost and downtime, use of resources effectively and supervision tools for IT organizations. Expansion often started with distributed computing as business tends to expand from one location to multiple locations across the globe, with a requirement to enhance the client-server model to incorporate distributed data from multiple locations. With the implementation of server based data-centers, many major issues such as data management, data consistency, storage medium, security and accessibility have eventually become main points of concern.

## 1.1 Cloud Architecture

As rapid data growth raised concerns in late 90's for enterprise environments, storage moved outside the servers (such as EMC, HP, Hitachi etc. storage appliances) to individual hardware units in the form of storage appliances that provided a significant solution for large scale data management [27]. Hypervisors were also created and used to make a virtualization layer on top of the operating systems. Integration of virtual machines (VMs) in the virtualization layer to the operating system environment added a robust and effective means to share the CPU load and processing time between VMs [13]. It was at this time that Cloud computing emerged.

Cloud computing is a paradigm shift in how information is stored and shared on a distributed platform in the form of VMs to provide services to customers on-demand. Due to its service oriented architecture, it has been defined as a "pay-as-you-go" model where customers use resources when required and pay for the used resources. A very general view of Cloud computing has been given by Buyya et al. [9]:

"I don't care where my servers are, who manages them, where my documents are stored, or where my applications are hosted - I just want servers always available and to be able to access them from any devices connected through the Internet; and I am willing to pay for this service for as long as I need it."

The ongoing success of Amazon, Microsoft and other public Cloud service providers induced leading technology companies like Oracle, HP, IBM, Adobe etc., to introduce their own Cloud computing platforms. By introducing their Cloud services tools, they were able to expand their service areas and business. However, when it comes to financial, health and defense services, public Clouds have several weaknesses, including

security and privacy issues, and this is where private Clouds become attractive.

Private Clouds have shown to be able to provide effective solutions for security and privacy issues of business and hence many other organizations have jumped into using private Cloud systems [41]. A private Cloud is more secure as the organization builds their own infrastructure and uses their own data storage servers. Private Cloud owners may also outsource their requirements like applications to a third party but underlying resources such as servers, storage and network remain non-shared.

In public Clouds, the Cloud provider controls the IT infrastructure of an organization and they ultimately have access to all data that is hosted or stored on the running virtual machines. Even though service level agreements (SLA) are in place to define the terms and conditions for data storage and access management, this central control of data by the Cloud provider and privacy concerns are too much risk for certain organizations. For example, government and military agencies do not consider public Cloud for their operations such as computing or storing their sensitive data [35][40]. A security breach at a Cloud data centre for such organizations can expose sensitive information to the outside world, which might bring a greater loss for a government and its citizens. To better understand these challenges, I have built a private Cloud.

## 1.2 Service Models in Clouds

Cloud computing is an evolving profit driven technology for IT Infrastructure business. It has provided financial freedom to clients by introducing a popular pay-as-you-go model. Integration of security features by Cloud providers has given an

additional edge to this technology. However, optimization of Cloud computing technology to provide an effective and cost beneficial solution in all application areas is still in progress.

Infrastructure as a Service (IaaS) is one of the service models of Cloud computing, that enables an organization to distribute its workload by outsourcing major data-center operations such as networking, storage, servers and analytical tools. Cloud providers are responsible for the hardware used to run operations and maintain them. Software as a Service (SaaS) is another service model, where applications are hosted by Cloud providers and made available to customers over a secured communication channel. SaaS provides an effective way of supporting on-demand software solutions for the end user. In the Platform as a Service (PaaS) model, customers can easily use the platform without implementing the infrastructure to run the applications. These models are available in public and private Clouds but security remains major point of concern for the organizations.

Security is one of the most important aspect of Cloud [1]. In private Cloud, virtual machines are hosted and managed in a company's infrastructure. This flexibility gives easy access and secure medium for data storage. Moreover, on-premise solution holds some advantages over Cloud infrastructure like more secure, easy operations on hardware and full control on infrastructure. These advantages of on-premise solution can be used in private Cloud with service flexibility of Cloud as well. As an added advantage private Cloud can be integrated with public Cloud, which is called as Hybrid Cloud. In case of Hybrid Cloud, third party VMs (public Cloud) can be added to private Cloud within premise of enterprise, this add to available resources

[1]IEEE Computing edge, special issue on Cloud Computing, page (34-40), September 2015

in case of more resources are required.

## 1.3 Problem Description

Implementing a private Cloud is a complex and cumbersome process for system analyst and administrators. Association of various components to make it service oriented architecture adds complexity in design. The Primary objective of my work is to implement a private Cloud and understand the implementation process. After implementation, my objective is to test the performance with respect to various parameters such as response time, replication status, data integrity and access on a data critical and time constrained application. For my thesis, I set the following as important steps in building a private Cloud that would satisfy the above objectives:

1. Implement open-source software Openstack on (Dell) servers to create a prototype private Cloud to achieve goal of creating VM on demand as a service;

2. Deploy a financial option pricing techniques on the private Cloud for application testing;

3. Data replication provide backup for data and system therefore explore strategies for data replication by -

   (a) Proposing a strategy to use object and block storage available in Openstack for replication in private Cloud with assessment of preparation delay time and data consistency as objectives.

   (b) Testing replication of data generated from financial application (option

pricing problem) using block level and object level storage and compare these replication strategies;

4. Analyse performance of VMs in my private Cloud on the basis of response time on the VMs by using financial application. Compare performance with standalone servers as well.

For my thesis I have built a private Cloud and performed experiments on it with a real-world financial application to evaluate my private Cloud.

# Chapter 2

# Cloud Computing Standards

## 2.1 Market Oriented Standards for the Cloud

Cloud providers are responsible to provide services in the form of computing resources like VMs to customers. To maintain the quality of service between Cloud providers and customers, specific Quality of service (QoS) is required to achieve objectives and keep operations functional [33]. A service level agreement (SLA) is agreed by an individual customer and the Cloud provider stating the required QoS parameters [35]. To implement this SLA, the Cloud provider may follow the Information Technology Infrastructure Library (ITIL) approach[5] that sets standards for the service request from customers. ITIL is based on service management focusing change, task and problem management. ITIL provide the freedom to Cloud provider to prioritize the service requests on the basis of QoS, market incentive and urgency. This differentiation of service request can be categorized on the basis of service utility (importance of service request) as well [10].

An SLA is maintained between a customer and Cloud provider, which includes all the details of service provided to customer and the penalties if services are not fulfilled. In case of any breach, the customer has full right to penalize Cloud service provider with agreed financial restriction mentioned in SLA . These breaches include downtime of any server and application running on them, slower response time than agreed to, unavailability of resources as per the agreement and any major failures which hamper the production services [35].

To improve the production services, it is important to automate the processes involved in providing the services. To automate processes, service orchestration is implemented by IT infrastructure companies. Service orchestration is an important aspect to serve the purpose of full operational ability along with the effective use of resources [34]. To streamline this aspect of SLA, IT services and policies are defined by the Cloud vendors with the agreement of the client. The use of SLAs help a Cloud solution to optimize the input-output process for the client with aggregation of a set of protocols that need to be followed to get the required services. The SLAs can be used in private Cloud as well if their are multiple departments in an organization and there is need for resource auditing and regulation.

## 2.2 Trends in Compute Utility Services - Public Clouds

Cloud services offer an efficient budget policy that have empowered organizations to implement software and infrastructure to use platforms. To fulfill their require-

ments for implementation of software, infrastructure and platforms, some major enterprise organizations have jumped into the Cloud market and the most prominent one is Amazon. There are many public Cloud providers such as Amazon, Microsoft, Google, IBM, Rackspace, HP etc. Amazon has created the Amazon web services (AWS) tool which encapsulates all the Cloud capabilities and provides flexibility to customers to use Cloud services as per their requirements [10]. This has led to rapid growth of Cloud adoption in big and small scale enterprises. one of the leading survey agencies stated [16]:

"New Morgan Stanley research expects AWS to hit $24 billion in revenue by 2022 and to put the hurt on legacy IT providers in the process".

There are predictions about how Cloud computing will bring change to computation process in companies. For example, Merrill Lynch Research notes [10] "Cloud computing is expected to be a $160 billion addressable market opportunity, including $95 billion in business and productivity applications, and another $65 billion in online advertising". Of particular note, the Cloud computing economy is leveraging the power of multiuser capability using the Internet and enabling the ability to share storage infrastructure between multiple users, as well as delivering extremely fast shared storage through interconnects at significantly reduced costs.

## 2.3 Market solutions - Private Clouds

The introduction of **open source** Cloud computing software platforms has provided advancements in Cloud service deployment. Now, small businesses are using private Clouds [19] based on these open source platforms to satisfy their needs of

elastic demand for resources and high-end computing business requirements. Elastic demand is the term used by Cloud vendors for dynamic provisioning of Cloud resources using workload sharing and VMs migration to other physical machines [20]. Open source software platforms such as Openstack [31], Cloudstack [11] and Eucalyptus [29] have made Cloud facilities freely available to researchers. One the open source software is deployed in organization's infrastructure it is made secure by creating private network. Organizations also have flexibility to modify open-source code according to their requirements.

Some mid-range and small businesses have opted for creating their own private Clouds using open source solutions such as Openstack, Cloudstack and Eucalyptus. All these are Cloud supported software platforms which set up the working Cloud environment on privately owned infrastructure. As the open source community is gaining popularity day by day, these Cloud solutions are also beginning to build a market of their own. Corradi et al. [14] and Ahmad et al. [3] discuss VM consolidation and VM migration in Openstack which bind together network resources, power, CPU time, fault tolerance and system maintenance to obtain efficiency in private Clouds. Private Cloud software solutions require a minimum running environment for setup which includes three components - Controller node, Compute nodes and Network node [14] (explained later in Chapter 4.2).

Wen et al. [42] compared Openstack and OpenNebula [28] on the basis of characteristics like "provenance, architecture, hypervisors, security and others in detail". They also provide recommendations for deployment of these private Clouds. They explain how Openstack can be a substitute for Amazon as an enterprise solution due

to its operation ability and scalability in the form of Cinder block storage [42], which can add storage blocks as an additional storage capability to ephemeral (temporary) storage. They provide important information like how OpenNebula is used by research institutions but it is unable to serve at the enterprise level due to constraints like the absence of any bare metal (Physical machine without operating system) service. They also provide models for deployment that can be used for building a private Cloud. Use of heterogeneous operating systems such as Unix and Windows were also analyzed to measure the impact on overall cloud performance.

Saha et al. [36] described how a Eucalyptus based private Cloud can be used as a service portal where various pricing algorithms are used and compared to get highest profit for Cloud provider. The Service portal application act as a finance service for Cloud provider, which can help to generate a pricing model for public Cloud instances. It uses the three node architecture (Compute, Storage and Network node) to setup the private Cloud and then deploys financial application for finance as a service. They also described how image and storage nodes can be integrated in the Cloud environment for scalability and reliability.

Among the various problems in using Cloud services, a significant one is implementation and performance evaluation for access to and reliability of user critical data. IT infrastructure organizations typically have a central data repository that contains all the important data required for the proper functioning of the organization. Data allocation and then replication are an essential part of IT infrastructure. In a public Cloud environment data remains at different locations; VMs at different locations hold that data which can be migrated from one host to another host to meet

the requirement of load balancing [21]. To create a replication environment in Cloud infrastructure, these VMs need to be replicated and their state needs to be stored as well.

The advantages of private Clouds include:

1. In the case of a private Cloud, data will always reside in-house, so security concerns are reduced; while on the other hand security is a significant issue in public Clouds as data lies on third party resources with the Cloud vendor.

2. An SLA is a document that captures services to be rendered by the Cloud provider to a client. All the features of an SLA such as VM failure recovery, monitoring etc., cannot be provided by public Cloud; public Cloud providers cannot follow best standards for VM placement as more than one customer can use the same VM. For example, a VM can be used by one company in the morning hours and other company use the same VM in the evening times for maximum utilization of resources. Public Cloud may have problem of dedicated VM allocation due to following the standard policy of maximizing resource utilization. Public Cloud providers sometimes use the same resources as explained above which can violate organization policies.

3. In a private Cloud compliance standards and applications are customer driven, whereas in the case of public Cloud, all application resources are configured by Cloud vendor, which acts as a constraint for the customers to override.

## 2.4 VM failure Recovery in Private Clouds

"VM failure recovery (DR) is defined as the use of alternative network circuits to re-establish communications channels in the event that the primary channels are disconnected or malfunctioning, and the methods and procedures for returning a data center to full operation after a catastrophic interruption (e.g., including recovery of lost data)" [1]. VM failure recovery involves restoring the technology aspect of a service. In 1978, Sun Information Systems was the first American company to provide a hot site to organizations [12]. A hot site is a facility that an organization can relocate to in the event of a system failure due to a natural VM failure or a human induced VM failure to resume the operation of a service.

As businesses became more reliant on their IT infrastructure during the late 20th century, there was pressure for organizations to have a VM failure recovery plan in place in the event a service becomes unavailable. As a result, different mechanisms were being provided as solutions. It has been statistically proven (see [12]) that spending one dollar on VM failure recovery planning will save you four dollars in the long run when trying to recovery from a VM failure. An important step in VM failure recovery planning is to identify which services are considered vital for an organization, how long can they afford to keep the service unavailable and identify which systems are associated with providing the service [12].

There are multiple strategies in designing an efficient VM failure recovery plan, here are a few strategies: backups of systems and data can be made and transported to an offsite location; a second alternative is to only forward data to an offsite location instead of copying the entire system; a third, more recent, alternative is to use a

private Cloud to replicate virtual machines, disks, templates and store them in the public Cloud; a fourth alternative is to use a hybrid Cloud to back up the data stored onsite and offsite and in the event of a VM failure launch the system from within the public part of hybrid Cloud.

## 2.5 Thesis organization

Private Cloud has been implemented and used by many enterprise organizations for commercial purpose. We use Cloud services in our daily life while storing images, documents and other files but what technology is used in back-end remains abstract. To understand this abstract idea I have implemented a private Cloud for my thesis in a small-scale testbed environment by deploying Openstack on Dell Servers. Most likely this is the first attempt to built a private Cloud in Computer Science department using Openstack.

Rest of the thesis is organized as follows: A primer on the option pricing problem is presented in Chapter 3. For my thesis I have built a Openstack-based private Cloud as described in Chapter 4. I proposed a strategy for replication of data from within the Openstack and show significant improvement, which are presented in Section 5.1. I executed one financial application, option pricing problem, that is data sensitive to clients and results are presented in Section 5.2. Performance analysis of my private Cloud virtual machines in comparison to standalone server is presented in Section 5.3. I conclude this thesis study in Chapter 6 by summarizing my work and considering possible directions for further work.

# Chapter 3

# Option Pricing Problem - A primer

For my work I have used a finance problem, which is one of the most suitable applications (due to security and privacy concerns) to experiment with on the private Cloud as any financial investor requires their financial portfolio secure and private. I have learned about option pricing problems which taught me the breadth and depth on finance markets and how Computer Science (algorithms and high performance computing) can help to solve some of the problems in finance using mathematical models. I have developed an application for the option pricing problem.

A Financial contract is an agreement between two individuals, groups or organizations to buy or sell any underlying financial asset. There are many financial contracts used in finance markets such as options, swaps, etc;. For expedited solutions, algorithms and programming practices have been used to solve problems in finance. Private Cloud is one of the most suitable platforms that can be used for financial applications to deploy and generate results.

In my work, I have studied one of the most commonly used financial contracts

called an option. An option is a financial contract where the option seller (writer) of the option writes an agreement on an underlying asset (for example, a stock) with the option buyer (holder) of the option. In an option contract, the holder has the right, but is not obligated to buy/sell the underlying asset during the contract period. An option contract is set for a certain period of time as per the agreement. Finding the worth of an option contract is known as the option pricing problem. Option pricing is considered one of the computationally intensive problems in finance which could use high performance computing algorithms to solve the problem.

## 3.1 Types of Options

There are two types of options:

1. Call Option - It gives the holder the right to buy an underlying asset at a predetermined price, without any obligation to exercise that contract on or before expiration. A call option has an expiration date. Depending on the terms of the contract, the underlying asset can be purchased on the expiration date only (European Style) or any time prior to the expiration date (American Style).

2. Put Option - It gives the holder the right to sell an underlying asset at a predetermined price, without any obligation to exercise that contract on or before expiration. A put option has an expiration date as well. Depending on the terms of the contract, the underlying asset can be sold on the expiration date only (European Style) or any time prior to the expiration date (American Style).

Figure 3.1 shows the process flow for a call option where call writer and holder

communicate using broker as middle-man. The Option Clearing Corporation (OCC) makes sure all the rules and regulations are followed between the writer and holder.

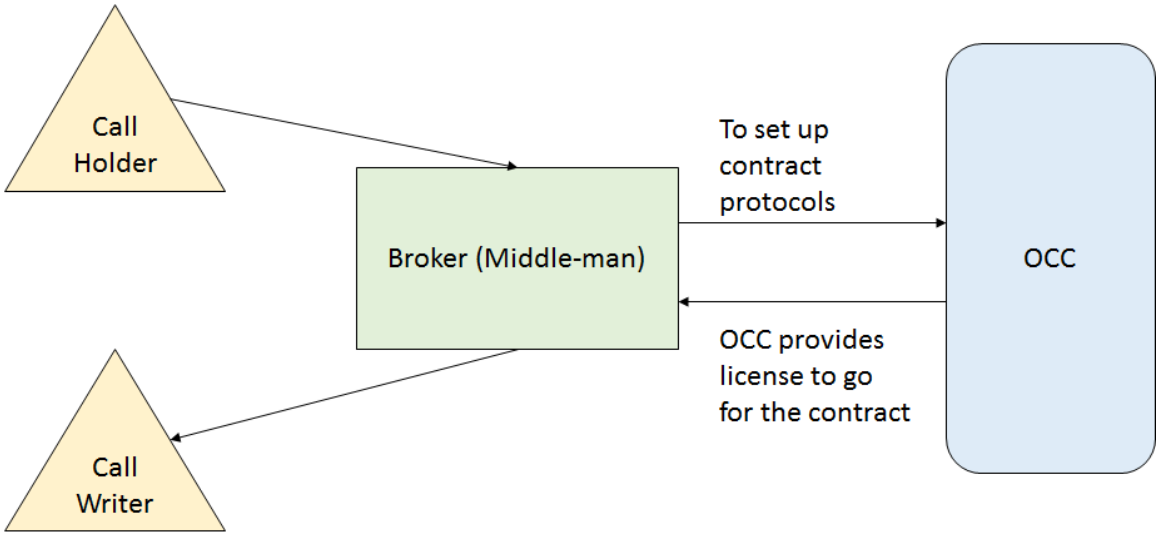



Figure 3.1: Process flow in call and put option.

## 3.2 Examples of European Call and Put Options

Let us suppose a customer is using European style of call option: The Strike price (S.P) of a stock or underlying asset is $100 and number of shares are 100. The Current price of each share is $98, the Option premium to get into the contract is $5 and the expiration date for the option contract is 4 months. Hence, the total investment that would be required at the time of execution of the contract is 100*100=$10,000. On the expiration date the customer has to decide if they want to exercise or not based on three scenarios: (i) if the current open market price is less than the strike price then there may be no exercise, (ii) If the current open market price is greater than

the strike price but less than the sum of the strike price and premium then there is no profit if exercised. The contract may still be exercised in order to get hold of the stock as it is something difficult to get otherwise; and (iii) if the current open market price is greater than the sum of the strike price and premium then there will be profit and it should be exercised.

Now, let us suppose a customer is using European style of put option; strike price (S.P) of a stock or underlying asset is $70 and number of shares is 100. The current price of each share is $65, option premium to get into the contract is $7 and the expiration date for the option contract is 3 months. Therefore, the total cash flow expected at expiry of the contract is 70*100=$7,000. On the expiration date, the customer has to decide if they want to exercise or not. There are three scenarios: (i) if the current open market price is greater than strike price than there may be no exercise, (ii) If the current price is less than strike price but greater than the sum of strike price and premium then there is no profit if exercised, and (iii) if the current price is less than the sum of the strike price and premium then there will be profit and it should be exercised.

## 3.3 Binomial Lattice

Black Scholes [6] and Merton [26] derived a closed-form solution to price the simple European style option. Pricing other styles of options led to the development of many numerical techniques. One of those techniques is Binomial lattice [15] described below.

Binomial lattice is a discrete-time option pricing model. It is used to compute option prices for both American and European styles. It is computationally inten-

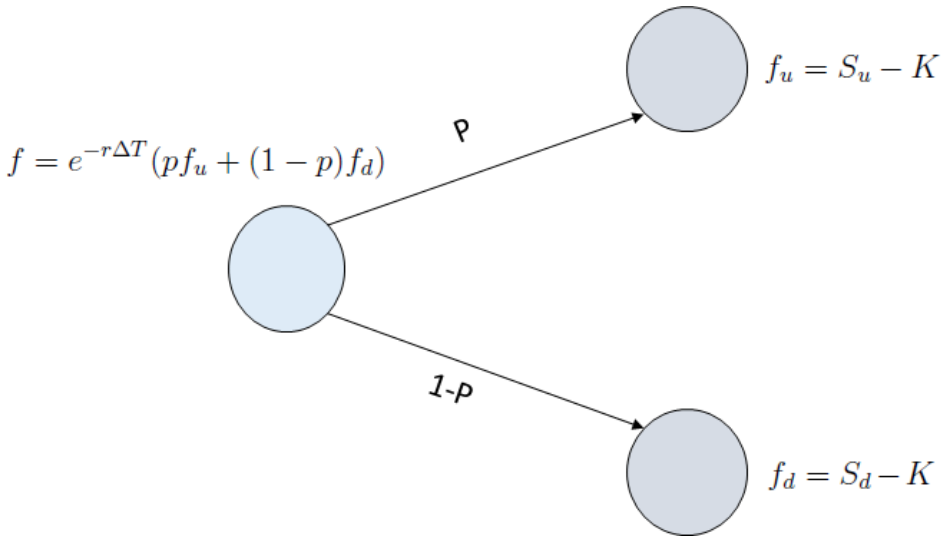



Figure 3.2: One Step Binomial

sive and sometimes requires large computing resources to solve the problem for high accuracy. Option pricing has two basic objectives: (i) determining the best time to exercise the option; (ii) finding the best option value for a better profit.

Figure 3.4 shows an example of a one step binomial tree, where $S$ is the initial value at the time of the contract period. S can go up by factor $u$ to $uS$ with the probability $p$. S can go down by factor $d$ to $dS$ with the probability *(1-p)* by the end of the contract period. The values $u$ and $d$ can be evaluated by using the following equations [15].

$$u = e^{\sigma\sqrt{\Delta T}}$$

$$d = e^{-\sigma\sqrt{\Delta T}} = \frac{1}{u}$$

As shown in the equations above, binomial Lattice for option pricing which involves a tree data structure where 'T' represents time, $\sigma$ represents market volatility,

$\Delta$ T represents difference in time between two steps and 'u' and 'd' represents the factor by which the stock price increases and decreases, respectively. First we build a tree with 'n' (number of branches) to compute the option value accurately. As 'n' increases, the accuracy of the results are expected to be better. Secondly, at each step of the tree, payoff is calculated for either call or put options.

$$f_n = Max[(S_n - K), 0], for\ call\ option$$

$$f_n = Max[(K - S_n), 0], for\ put\ option$$

Here, $K$ is the strike price and $S_n$ is the spot price (price of the asset at that particular node of the tree) for the underlying asset at the n$^{\text{th}}$ step. If the value of S moves up $f_u$ is calculated and if S moved down $f_d$ is calculated where $f_u$ and $f_d$ are the payoff at the next step. That is $f_u = S_u - K$ and $f_d = S_d - K$ as in Figure 3.4 for a call option. The current value of an option for a node with $e^{-r\Delta T}$ (discounting factor) is calculated using two branch node values by using the formula [15]:

$$f = e^{-r\Delta T}(pf_u + (1-p)f_d)$$

where,

$$p = \frac{e^{r\Delta T} - d}{u - d}$$

This computation is repeated at each step (in the reverse direction) until we reach the root node. The option value computed on the root node is the final result of this computation. The size of the tree is varied according to the required accuracy of the results prescribed by the user.

Binomial lattice problem for option pricing is used for application testing of the my private Cloud in Section 5.2.

## 3.4 Monte Carlo Simulation

Boyle [8] was the first to introduce numerical methods in option pricing using Monte-Carlo (MC) simulation in 1977. MC has become a popular method to compute the value of financial option value. Many scholars have contributed to the literature on MC simulation such as Hull and White [22], who used the MC approach to obtain more accurate option values than the Black-Scholes model. Similarly, Schwartz and Torous [37] used payment behavior and evaluated the results that closely relate to actual price. As the literature on MC grew, more work in computational finance [7][18][32] has been reported.

This is similar to the Binomial lattice technique explained earlier. The algorithm is given below:

**Algorithm 1** Monte Carlo pricing Algorithm

1. Initialize the parameters such as $S, r, K, T, \sigma$,
2. for $i = 1$ to $M$ do /*$M$ = number of simulations. i.e., for each simulation*/
3. for $j = 1$ to $n$
4. simulate sample paths of the stock prices.
5. next j
6. for each simulated path, compute the pay-off of the option.
7. next i
8. Compute the discounted average of above simulated pay-offs.

Boyle et al. [7] have divided the MC technique into three steps: Simulation of the sample path for underlying asset prices, evaluation of option value on each path using discounting and averaging the calculated discounted option value over sample

paths. After simulation of the sample path, payoff for each path is calculated. Each payoff value is evaluated using discounting factor to calculate the option value for each path. Averaging is done for all the sample paths to calculate final option value.

MC simulation experiment is used for performance testing of the Cloud VMs in Section 5.3.

# Chapter 4

# Private Clouds

There are multiple open-source software systems available to build a Cloud. Cloudstack is an open source software that allows for a Private Cloud and Hybrid Cloud deployment [39]. The main functionality of Cloudstack is to operate a large scale deployment of a virtual infrastructure by managing a large number of virtual machines. Eucalyptus is another open source system used to build Private Clouds and Hybrid Clouds that are compatible with Amazon Web Services [29]. Eucalyptus manages resources that allow for some dynamic allocation of resources. OpenNebula is another open source system that allows for the deployment of Public Clouds, Private Clouds, and Hybrid Clouds [28]. One of the important functionality of OpenNebula is to manage data that is distributed among data-centers and to ensure that these data-centers are able to exchange information with each other regardless of their infrastructure. Openstack is an open source software that is primarily used to manage a virtual infrastructure using the Dashboard (Horizon) for central management of all resources or the Openstack API.

| Characterstics | Cloudstack | Openstack | Eucalyptus |
|---|---|---|---|
| Architecture | Monolithic | **Fragmented, Distributed** | Five parts |
| GUI | Strong EC2 like | **Multiple CLI based** | Limited |
| Security | Low | **Strong token based** | High |
| Load Balancing | Low | **High** | Medium |
| Scalability | Low | **Very high** | Medium |
| Market spread | Good | **Very Good** | Fair |
| Hybrid Support | Not supported | **Supported** | Best integrated with EC2 |
| Installation | Medium expertise | **Difficult** | Medium level |

Figure 4.1: Open Source cloud software comparison

A tabular comparison is shown in Figure 4.1 where Cloudstack, Openstack and Eucalyptus [31] [11] [29] are compared on the basis of certain important Cloud characteristics. Openstack is better than other two when it comes to security as Openstack uses token based authentication feature of keystone which is not present in the other two. Openstack is also very scalable due to its Cinder component which provide additional block storage other than object storage. On the basis of these characteristics it can be said that Openstack is a standout performer in comparison to other two. The only drawback with Openstack is the typical installation process. Openstack installation and some of the major challenges that I faced while deploying Openstack are described in the Appendix.

An important feature that makes Openstack advantageous compared to the other options is that Openstack supports small scale deployment. Openstack can be tested using the development version called as Devstack that supports deployment onto a single, local machine for rapid application development and testing with minimal

required effort in setting up. For these reasons, I selected Openstack in my thesis work as the Cloud computing software to build a private Cloud and to enable the VM failure recovery solution.

## 4.1 Openstack

Originally Openstack was developed as a collaborative project between Rackspace and NASA [38]. In 2010, Openstack was released under the name Austin. Austin had very limited features, for example, it had support for object storage only initially. Companies started to contribute to Openstack because they began to see its potential in the virtualization market. Contributors added more features to Openstack, and eventually in 2012 support for Cinder was integrated [3]. Cinder provided a mechanism to create, access and manage volumes in Openstack. The main functionality of volumes is to store data a virtual machine uses. Volume storage provides block as storage medium which contains raw blocks to store data. Whereas Object storage stores data as objects where object contains data attributes and metadata. In recent years, Openstack has gained a lot of popularity due to its flexibility and ability to provide a virtualized infrastructure as it provides multiple hypervisors such as KVM, qemu and Hyperv.

Openstack provides a means for small companies to provide their customers with services without the need for a significant investment initially. It thus became easier for new service providers to start out who did not have a lot of investors. At the same time, Openstack provided a means for big organizations to implement private Clouds with reduced investments in physical hardware due to the ability to more

effectively share such hardware between VMs. Openstack is currently an open source project and has hundreds of contributors. What makes Openstack so advantageous is that if there is a need for a feature there are contributors constantly available to implement them and fix them if any bugs are discovered. Another reason Openstack is advantageous is that as an open source project it does not require any subscription or an annual fee to use.

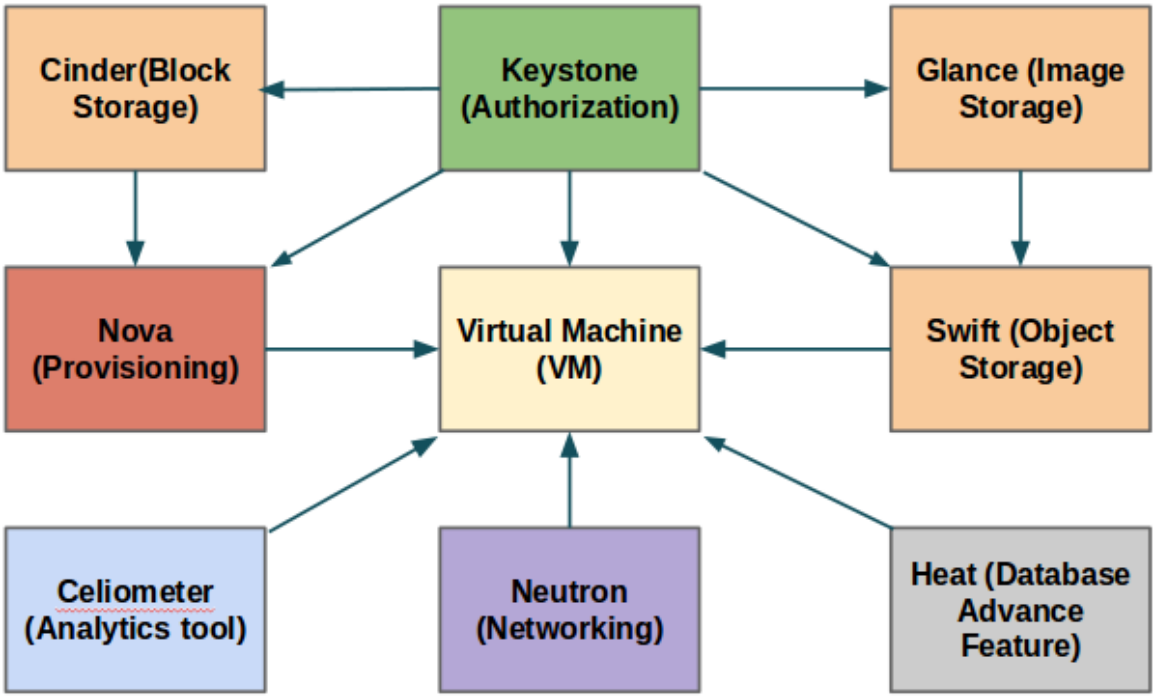



Figure 4.2: Internal component connection in Openstack.

Several components contribute in building an Openstack based Cloud as shown in Figure 4.2. Of the nine components, three components Swift, Cinder and Glance provide storage mechanisms for Openstack. Swift provides an object storage capability, Cinder provides a block storage capability, and Glance provides a repository to store the virtual machines a user creates in Openstack or downloads them from the

internet. Keystone authorizes the user to access the Horizon dashboard. It use token based security mechanism. Nova communicates with other components to provision resources for the virtual machines. Neutron does the networking infrastructure setup within all 3 Openstack node types (Controller, Compute and Network). Two of the storage components that are crucial to my thesis work are explained next.

### 4.1.1 Cinder

Block storage was a fundamental milestone for Cloud computing since it provides the capability to store virtual machines along with the data those virtual machines use. Before block storage was integrated into Openstack, virtual machines used object and ephemeral storage [38]. Virtual machines using ephemeral storage have a significant, fundamental flaw. The flaw is that when the virtual machine powers down, all of the data and contents of the virtual machine are lost. This is a problem because a virtual machine may contain important data required to provide services to customers. Consider a banking transactions service. If the virtual machine powers down, all of the customers banking information is lost. The organization will not know what the last balance on the customers account was if that information was not saved to persistent storage. Simply powering on the virtual machine would not solve the problem because even though we have the environment to provide the service, we need the data of past transactions. Finding a solution to ensure a virtual machine never powers down is not a realistic answer to the problem.

There is a need to provide a mechanism that allows data to persist in the event that a virtual machine powers down. Block storage provides one possible solution to

this problem. The data persists within the volume even though the virtual machine may power down and the data is available the next time the virtual machine powers up. Cinder was developed and integrated into Openstack to provide access to and management of a block storage created by a user.

Figure 4.3 shows the architecture of Cinder. Here, cinder API is responsible to interact with the user, which in response request to create volume or take volume snapshot. Cinder scheduler assigns volume and snapshot creation tasks to queue sequentially. Database (DB) holds the volume data and assign block address for future use. It also assigns unique block address for each volume and snapshot.

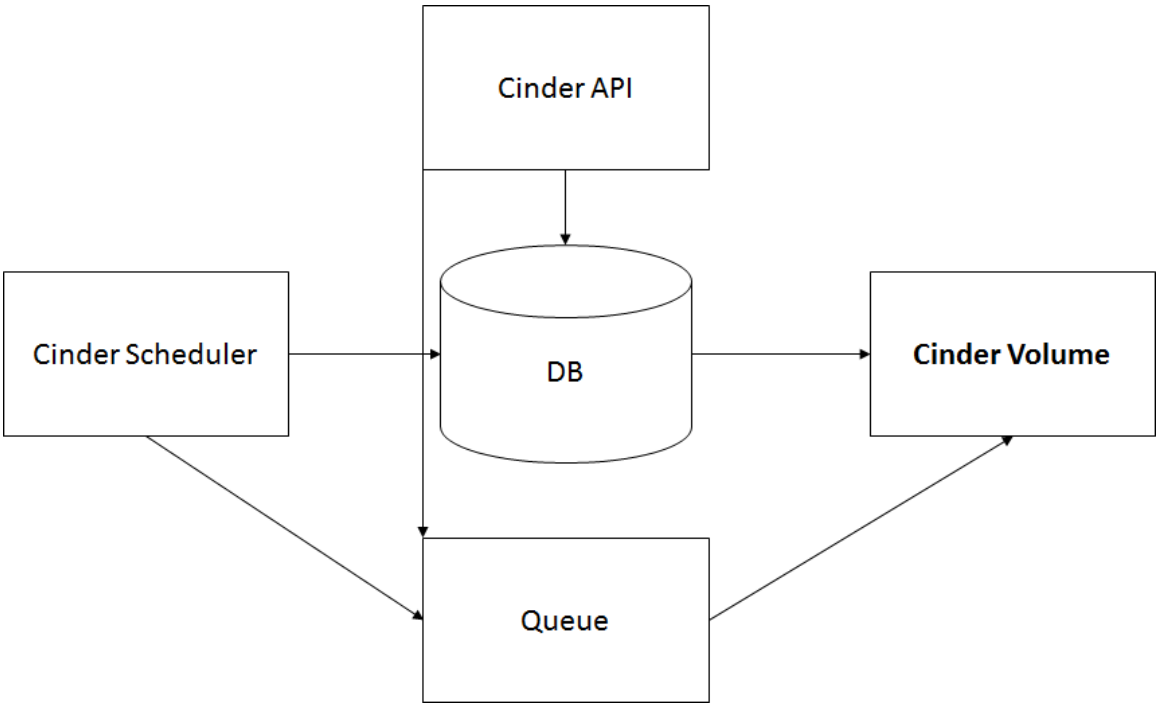



Figure 4.3: Architecture of Cinder

### 4.1.2 Swift

In Openstack, the module named Swift provides the object storage component in the application. Swift is used to implement something called an Openstack cluster. The Openstack cluster is used to provide a distributed object store. The object store uses HTTP protocols PUT to update an object or GET to retrieve the most recent version of the object. Swift implements something called an Openstack ring. This ring consists of a proxy server and a storage node. These two components are used to store and retrieve objects upon a user‘s request. The main functionality of the proxy server is to track the location of the most recent version of the object and to determine which storage server it should send the most recent version of the object. These Openstack rings can be divided in various ways. The reason for the division is if in the event a request for an object fails, then there is another entity available who can fulfill this request. Three common divisions of Openstack rings are by disk, object server and zone as shown in Figure 4.4.

Within each of these divisions the data is replicated in the event one of the providers has a failure so another entity can fulfill the request. Through various testing it was determined that maintaining three replicas was generally sufficient for reliability so by default Openstack maintains three replicas of the data [3]. Swift implements eventual consistency. This means that when an object is updated not all of the storage servers are updated immediately to contain the most recent version of the object. Only one of the storage servers receives the most recent version of the object from the proxy server and then the storage server propagates that object to the rest of the storage servers storing replicas of the object as a background task. A

risk with eventual consistency is that a storage server may receive an updated object but then may fail before it has a chance to propagate that updated object to another storage server. In a future updated versions of Openstack [24], code named Grizzly, allowed users to maintain as many replicas as they wanted within the Openstack ring.

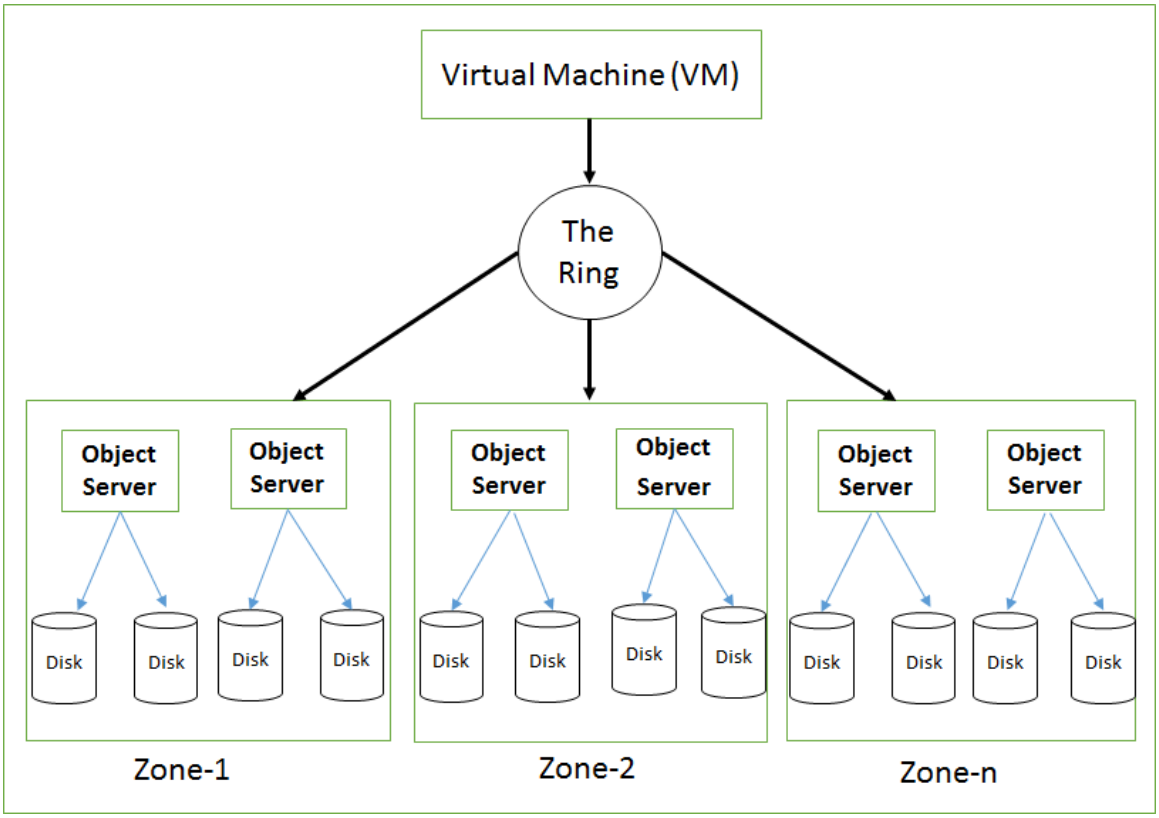



Figure 4.4: Architecture of Swift

Another feature implemented in Swift is time based sorting where the proxy server requests the most recently updated object from the fastest responding storage server. The main reason time based sorting was implemented was that storage servers were beginning to be distributed over larger areas, for example, the other side of the country leading to large latencies to access objects if a distant storage server was selected. A

major advantage Swift provided was that it allowed the capability to store objects on different platforms for example, Cleversafe, Scality, and Amazon S3 [5].

## 4.2 Building a Testbed

To build a private Cloud, I used three Dell R420 servers with multiple Ethernet ports. All servers have 4GB RAM and 8 Intel Xeon processors on each of them. Ubuntu 14.04 server was used as the operating system running on each of them.. I have used Openstack to create a private Cloud environment for these machines. Openstack is a better Cloud software in comparison to OpenNebula and Eucalyptus due to its properties such as load balancing, scalability, market spread and security as explained earlier (see for example, [42] and Figure 4.1). Openstack is a Cloud software platform with a three node architecture [31] as shown in Figure 4.5. Openstack should have minimum three nodes to implement Cloud but to get more resources more number of compute nodes can be added. There can be only one controller and network nodes in Openstack setup. The three node types in Openstack are explained next.

### 4.2.1 Controller Node

Controller node is responsible for running the basic Openstack services required for private Cloud environment to function. These node:

1. provide APIs, which are accessible by users for various functionalities.

2. run numerous services with high availability, utilizing components such as Pace-

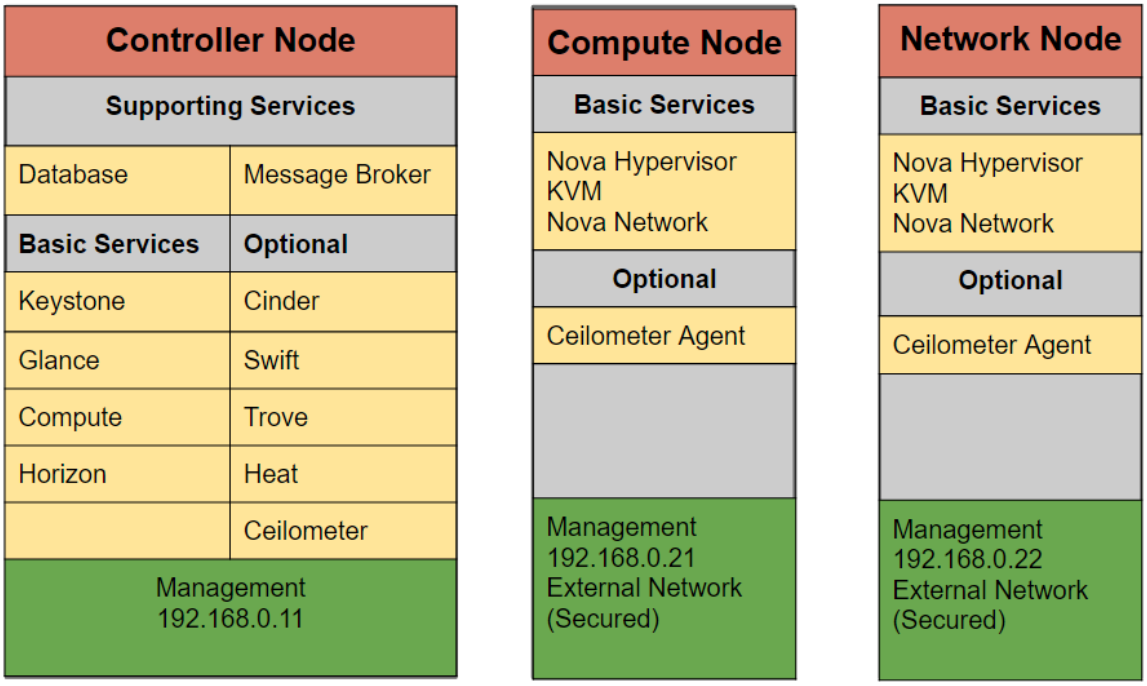



Figure 4.5: Architecture of Openstack based Cloud.

maker and HAProxy [17] to provide load-balancing and virtual server allocation functions.

3. provide always accessible "infrastructure" services, such as MySQL and RabbitMQ, combining the services using robust databases.

4. provide "persistent storage" through services run on the operating system. Backups of "persistent storage" is created on storage nodes for reliability purpose.

### 4.2.2 Compute Nodes

Compute nodes host the VMs in an Openstack environment. They:

1. execute the minimal set of services required to run these VMs.

2. use locally defined storage on the nodes for the VM so that recovery, in case of node failure, is possible.

3. execute the cloud users' code within the VMs they support.

### 4.2.3 Network Node

Network node provides communication channels and the virtual networking required for users to build public or private networks. Network node can be used to provide uplink to their VMs in the external network [31]. The network node:

1. creates the only ingress and egress points for access and security features of the Openstack running instance.

2. runs the environment's networking services other than networking API service.

## 4.3 Openstack components

VM provisioning is an important step for any Cloud. Below are the basic components required for the VM provisioning:

1. Block Storage (Cinder) - "Provides persistent block storage for running instances" [31].

2. Object Storage (Swift) - Uses a RESTful, HTTP based API to retrieve unstructured data objects and operating system cloud images.

3. Network (Neutron) - Enables the communication channel for other Openstack services, such as keystone, nova etc.

4. Authorization(Keystone) - Is an authorization and authentication service for other component of Openstack.

5. VM Image (Glance) - VM images are stored and retrieved by this service.

6. VM Provisioning (Nova-Compute, KVM hypervisor integrated)

7. Controller provisioning (Nova-Controller)

Openstack provides all these as separate but inter-related components as shown in Figure 4.2. These components help to establish services within the framework.

## 4.4 Implementation

The VMs on the compute nodes in my testbed are working on arbitrary computation algorithms that could be memory intensive. Openstack provides snapshot technology, I have used this technology to capture the state of VM images and these snapshots are scheduled to be triggered in a timely manner [30]. Along with the snapshots, I initiated block replication on the VMs, which captures the raw data on the data blocks and stores them on other passive systems (replica of live VM which is in offline mode) that are created and kept in idle mode. These data blocks can be retrieved in case of VM failure and provide same data to an active node. With image snapshot and synchronous replication, I show that (as explained in next Chapter) that data is more reliable.

# Chapter 5

# Experiments, Results and Evaluation

After the setup of virtual machines in the form of instances in Openstack, dynamic IP allocation (for internet access) is done followed by testing for the connection using Address Resolution Protocol (ARP), Java runtime environment and MYSQL are deployed on all the nodes for multiple operations. After the setup of all three nodes (one of each type) using Openstack with VMs, I run arbitrary computation tasks on the multiple VMs hosted on the compute node. The snapshots are captured pre-defined schedule using scripts. After testing is performed using a financial application, where one of the VMs is triggered with configuration errors (edit instance and increase memory size more than physical machine) that is followed by powering off that VM. At that point, snapshots for that VM are retrieved and passive data blocks are activated. This produces a replica of current VM in the powered off state. With this method, I am able to test the reliability of time constraint and data critical applications. Time

constraint applications are those that are required to be up and running all the time since any downtime can bring severe loss to the business.

## 5.1 Replication strategy evaluation

In the VM failure recovery experiments, if the data can be retrieved using snapshot and volume replication technique then we have a model for VM failure recovery within the Cloud. On the basis of data retrieval following the VM failures, I also analyzed the best (between Swift and Cinder) replication strategy that can be used in the Cloud environment. A typical comparison of snapshot and volume replication is part of the evaluation process to find the best VM failure recovery solution.

In my test scenarios, four VMs were deployed in the Cloud. Three of them were virtual machine instances and the fourth system was an instance being launched using a volume block containing the Ubuntu image. Of the three virtual machine instances deployed, two of them were clients and the third was a server. The fourth instance was used as a server as well, but deployed using a volume block. Cinder was used to take a snapshot of the instance deployed from the volume block and Swift was used to take a snapshot of the server virtual machine instance. After the systems were deployed, a simple RMI (Remote method invocation) application was developed and deployed on the instances to ensure they could communicate and transfer data between each other.

I created multiple shell scripts for the automation of the snapshot and recovery processes. These scripts were helpful to generate timely triggers to create and delete snapshots. I also created a script for data generation on the Ubuntu based Linux

system. Scripts that take snapshots of each instance at timed intervals and maintain a specified number of backups, were needed to perform the experiment. For object storage, there were three scripts created for taking a snapshot of the instance (multiple experiments). The first script maintained the iterations, decided which instance to target for the snapshot, and finally launched the other two scripts over every iteration that performed the actual snapshot. The second script deleted any old snapshots exceeding their lifespan and then created a new snapshot. The third script was used to force the data to be written into disk and prevent the instance from accepting input while the snapshot is taken to prevent any data inconsistencies as this was documented as best practice in the Openstack documentation [31].

For taking a snapshot using block storage, there were two scripts associated with the process. Again, one script was used to maintain the iterations, which volume to target, and to launch the other script at timed intervals. The second script would delete old snapshots exceeding their lifespan and then create a new snapshot of the volume.

Generating data for the experiment manually would have been a time consuming process. For this reason, a script was developed in order to create text files with readable information in order to have sample data during each iteration. Due to the time it took to create the script for generating the data, a snapshot was taken every two hours instead of every three hours.

| Iteration | Test-Mem | Mem-G | ST(sec) | VT(sec) | SD(sec) | VD(sec) | ST+SD(sec) | VT+VD(sec) | Efficiency |
|---|---|---|---|---|---|---|---|---|---|
| I-1 | 100 MB | 105.4 MB | 108 | 1 | 143 | 13 | 251 | 14 | 94.44% |
| I-3 | 300 MB | 309.8 MB | 151 | 2 | 150 | 13 | 301 | 15 | 95.01% |
| I-5 | 500 MB | 513.2 MB | 193 | 3 | 155 | 13 | 348 | 16 | 95.40% |
| I-4 | 1024 MB | 1050.06 MB | 368 | 5 | 233 | 13 | 601 | 18 | 97.00% |
| I-5 | 3072 MB | 3153.09 MB | 712 | 10 | 349 | 13 | 1061 | 23 | 97.83% |
| I-6 | 5120 MB | 5254.15 MB | 1205 | 15 | 470 | 13 | 1675 | 28 | 98.32% |

Table 5.1: VM preparation delay and efficiency for Cinder and Swift

### 5.1.1 Snapshot creation and VM deployment

Table 5.1 I observe which methodology, object storage or block storage, is faster in terms of the time it takes to create a snapshot, the time it takes to deploy a snapshot, and the amount of data associated with each iteration. Here, Test-Mem denotes Test Memory used as data block size, Mem-G denotes memory generated after taking snapshot, ST refers to time taken to create VM snaphsot (in seconds), VT is time taken to create volume snapshot (in seconds), SD and VD are the time taken to deploy VM and volume snapshot respectively. These numbers are captured in Figure 5.1.

Analyzing the results from Table 5.1 and Figure 5.1, in terms of creating a backup, and deploying the backup, block storage (Cinder) is seen to be much faster than object storage (Swift). For example, with test memory of 5120 MB, Swift requires 1675 seconds while Cinder requires only 26 seconds, leading to a relative efficiency of 98.32% (((1675-28)/1675)*100). Also, note that the speed of Cinder increases with the data size giving the opportunity to handle big data. As shown in Figure 4.3 and 4.4, Swift snapshot data is copied to multiple disks using object server which makes the process time consuming, whereas in Cinder, the snapshot of a volume is taken and stored as a single copy. This is one of the reasons for this high relative efficiency with

Cinder, there can be other factors which affect replication process such object server assignment and ring scheduling for object server allocation. For an organization, this is a significant factor to consider when creating a VM failure recovery plan because they want to make backups quickly to minimize overhead on the system so there is minimal interference with service quality. The system can be recovered a lot quicker using Cinder instead of Swift which means the system will be down for shorter periods of time.

As more data is stored on these systems we may naturally expect the time to create a backup and backup deployment to increase. Also, from Table 5.1 I observe that the speedup is consistent with data size. That is, if it takes one minute to deploy a backup using block storage, then we can expect a deployment using object storage to take approximately 64 minutes (for a data size of 5 GB). This is a significant difference and having a service down for approximately 64 minutes would likely not be acceptable to an organization that relies heavily on their IT infrastructure. Therefore, in terms of recovering a system from a VM failure, it is clear that using block storage (Cinder) is the better option.

### 5.1.2 Data Consistency

Table 5.2 shows which methodology, object storage or block storage, is better at preserving consistent data by measuring how many bytes of data are lost at each iteration. In this table, V-snap refers to bytes in VM snapshot, Lost-VM denotes bytes lost in VM snapshot after deploying as new VM, V-VM refers to bytes in volume snapshot and Lost-V as bytes lost after deployment of new VM using volume

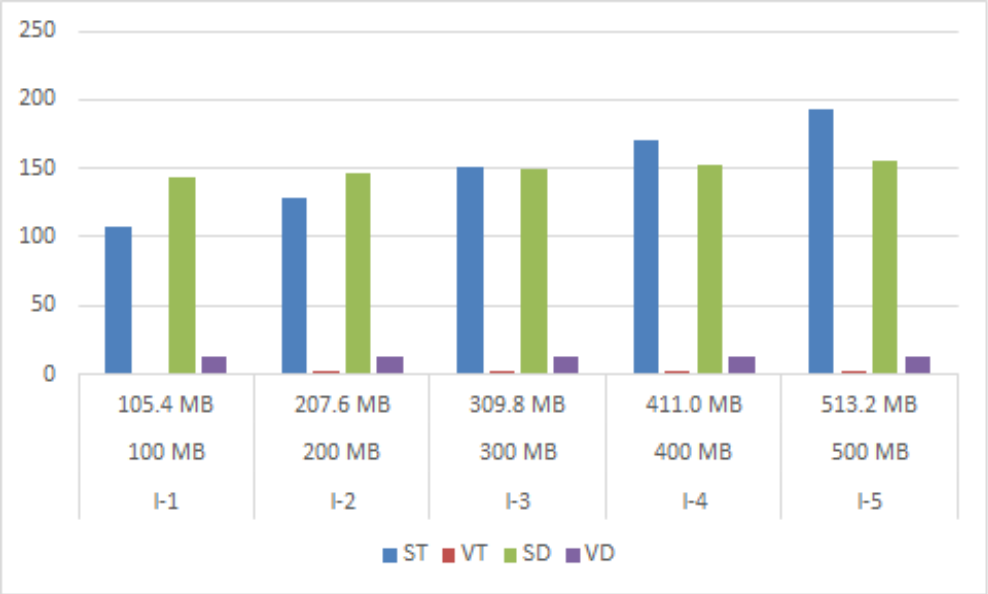



Figure 5.1: VM preparation delay analysis

snapshot. These results are captured in Figure 5.2

| Iteration | Total-Bytes | V-Snap | Lost-VM | V-VM | Lost-vol |
|---|---|---|---|---|---|
| I-1 | 10538598 | 10538598 | 0 | 10538598 | 0 |
| I-2 | 20762214 | 20762214 | 0 | 20762214 | 0 |
| I-3 | 25480806 | 25480806 | 0 | 25480806 | 0 |
| I-4 | 15257190 | 15257190 | 0 | 15257190 | 0 |
| I-5 | 26929875 | 26929875 | 0 | 26929875 | 0 |

Table 5.2: VM recovery data

Analyzing the results in Figure 5.2, we can observe that both components, Cinder and Swift, experienced no data loss. As the amount of data increases on each system, it would be expected that Swift and Cinder would begin to experience some data loss due to time lapse of preparation delay. Based on the results, I can conclude that both Cinder and Swift are effective in preserving small sets of data.

In future experiments, bigger sizes of data in the order of 20 GB would be used to

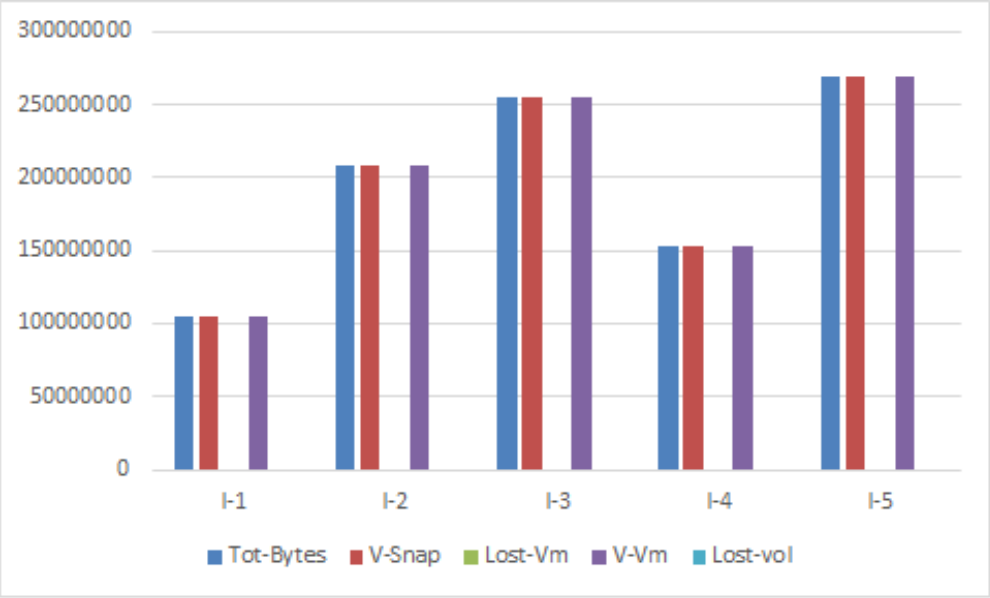



Figure 5.2: VM data recovery analysis

test how much data we can capture in snapshot before we experience data-loss from using Cinder and Swift. We would expect to lose data when more data is stored on the system with Swift or Cinder but the goal would be to determine which component provides more consistent data when the data gets large. As I expect that more data will increase the preparation delay time, which will leave the window open for new data to be written on the virtual machine and hence snapshot would not able to capture the latest data on the VMs. In terms of providing consistent data, for this experiment, both Swift and Cinder performed equally well.

### 5.1.3 Observations

Based on the results from Figure 5.1 and 5.2, I conclude that if the main objective of a VM failure recovery is to restore service as quick as possible, then Cinder would be

the best choice. I also conclude that both components function with equal reliability if our system is small. At the beginning, all systems begin relatively small, therefore, if we have two choices that both provide equal reliability and Cinder being faster than Swift, then Cinder would be the better choice at that particular point in time. As the system grows, more experiments could be performed to help determine which component provides more consistent data as the size of the data grows and determine the trade-offs if any.

Note- Some of the text for Private Cloud and Replication testing is taken from my publication [4].

## 5.2 Financial application deployment and observations

In the application testing experiments, I checked if the VMs created in the private Cloud are able to withstand application load and generate results on user request so it can be used as a SaaS. After the application deployment, I also analyzed the response time from the Cloud VMs. All the implementation is done on the Cloud VM level and applications were created using the Java platform.

There were five VMs deployed in the private Cloud. Four VMs (EuropeanCall, EuropeanPut, AmericanCall and AmericanPut) were used for the binomial lattice problem where Binomial put and call options were used for European and American styles (Client systems). I have divided implementation considering the request to get option values are from many users than different VM can respond to that request.

To fulfill that, One of the VMs (Server) was used to deploy code from where all the classes were called on the client systems. Connection between VMs were done using the server VM. After the systems were deployed, a simple RMI application was developed and deployed on the instances to ensure they could communicate and transfer data between each other similar to replication testing process.

The configuration used to create the Cloud VMs were the same for all five VMs. The Ubuntu Cloud image is used as operating system, 512 MB RAM and 2 virtual CPUs are set as a configuration parameter. IP association is done to provide Internet access and intercommunication on all the VMs.

Java code created for this problem was deployed on the server machine and each VM is responsible for its assigned problem. EuropeanCall VM was deployed for Java code of European call and likewise for EuropeanPut, AmericanCall and AmericanPut. Server has the main class which will invoke the user defined choice menu among the four VMs. On the selection of each one of the input parameters from above mentioned VM choices, activate the respective VM, which in response generates a user input values for stock price, strike price, rate of interest, time (no. of years) and number of steps. After the values are entered by the user, VMs generates the output in the form of option value. Multiple requests can be processed simultaneously by the server VM.

For each experiment, I have used several iterations to get multiple values for different set of inputs. Table 5.3 shows the results for the European call where the option value is computed on the basis of a set of values given by the user for stock price, strike price, rate of interest, volatility, time (no. of years). Similarly, Table 5.4 shows the results for the European put, Table 5.5 for American Call and Table 5.6

for American Put.

| Experiment | Stock price | Strike price | Interest | Volatility | time (years) | no. of steps | option value | BSM |
|---|---|---|---|---|---|---|---|---|
| I-1 | 50 | 40 | 5% | 20 | 0.5 | 11 | 11.091 | 11.087 |
| I-2 | 50 | 50 | 5% | 20 | 0.5 | 11 | 3.374 | 3.444 |
| I-3 | 50 | 60 | 5% | 20 | 0.5 | 11 | 0.447 | 0.511 |
| I-4 | 400 | 350 | 12% | 30 | 1 | 22 | 101.536 | 101.227 |
| I-5 | 600 | 600 | 12% | 30 | 1 | 22 | 107.314 | 106.701 |
| I-6 | 600 | 650 | 12% | 30 | 1 | 22 | 82.130 | 82.507 |

Table 5.3: Results for European Call

To test the accuracy of my option value results, I calculated the option value using only closed-form solution given by the Black-Scholes-Merton (BSM) [23] formula for European Call. For example, the option value obtained (11.087) from the BSM formula for European call with the same parameters as in experiment I-1 in Table 5.3 is close to the option value that I got (11.091) using Binomial lattice approach for European call and similar results are observed for other experiments. Therefore, I can say that the binomial lattice implementation on Cloud VM is successful as it produces accurate results when compared to BSM model.

| Experiment | Stock price | Strike price | Rate of Interest | Volatility | Time (no. of years) | No. of steps | option value |
|---|---|---|---|---|---|---|---|
| I-1 | 50 | 40 | 5% | 20 | 0.5 | 11 | 0.103 |
| I-2 | 50 | 50 | 5% | 20 | 0.5 | 11 | 2.140 |
| I-3 | 50 | 60 | 5% | 20 | 0.5 | 11 | 8.966 |
| I-4 | 400 | 350 | 12% | 30 | 1 | 22 | 11.959 |
| I-5 | 600 | 600 | 12% | 30 | 1 | 22 | 39.466 |
| I-6 | 600 | 650 | 12% | 30 | 1 | 22 | 58.628 |

Table 5.4: Results for European Put

Application deployment is a complex process for a financial investment organizations that would like to keep their data secure and more agile. This prototype implementation could be used in financial organization to secure their data privacy.

| Experiment | Stock price | Strike price | Rate of Interest | Volatility | Time (no. of years) | No. of steps | option value |
|---|---|---|---|---|---|---|---|
| I-1 | 50 | 40 | 5% | 20 | 0.5 | 11 | 11.091 |
| I-2 | 50 | 50 | 5% | 20 | 0.5 | 11 | 3.374 |
| I-3 | 50 | 60 | 5% | 20 | 0.5 | 11 | 0.447 |
| I-4 | 400 | 350 | 12% | 30 | 1 | 22 | 101.536 |
| I-5 | 600 | 600 | 12% | 30 | 1 | 22 | 107.314 |
| I-6 | 600 | 650 | 12% | 30 | 1 | 22 | 82.130 |

Table 5.5: Results for American Call

| Experiment | Stock price | Strike price | Rate of Interest | Volatility | Time (no. of years) | No. of steps | option value |
|---|---|---|---|---|---|---|---|
| I-1 | 50 | 40 | 5% | 20 | 0.5 | 11 | 0.105 |
| I-2 | 50 | 50 | 5% | 20 | 0.5 | 11 | 2.293 |
| I-3 | 50 | 60 | 5% | 20 | 0.5 | 11 | 10.000 |
| I-4 | 400 | 350 | 12% | 30 | 1 | 22 | 13.691 |
| I-5 | 600 | 600 | 12% | 30 | 1 | 22 | 47.172 |
| I-6 | 600 | 650 | 12% | 30 | 1 | 22 | 73.343 |

Table 5.6: Results for American Put

Financial institutions prefer the traditional data-center architecture for their internal use. Application testing implemented for my thesis can be similarly deployed within an on-premise private Cloud in any financial institution.

These results show that private Cloud can be used to deploy any kind of application with intercommunication among the Cloud VMs. This application test also helps to understand the mechanics of a private Cloud for a real-world application. I may also use this application as a service for the investor (who may not be aware of advanced algorithm of option pricing) in the form of software-as-a-service (SaaS). Saha et al. [33] used the Eucalyptus Cloud to create the SaaS based financial application. My work is different as it uses Openstack based private Cloud and explores the system level features of private Cloud in comparison to application level features. Also, as already explained in Chapter 4 (refer figure 4.1), Openstack has various advantages over Eucalyptus.

Figure 5.3 depicts a possible SaaS model for financial institutions. If the investors

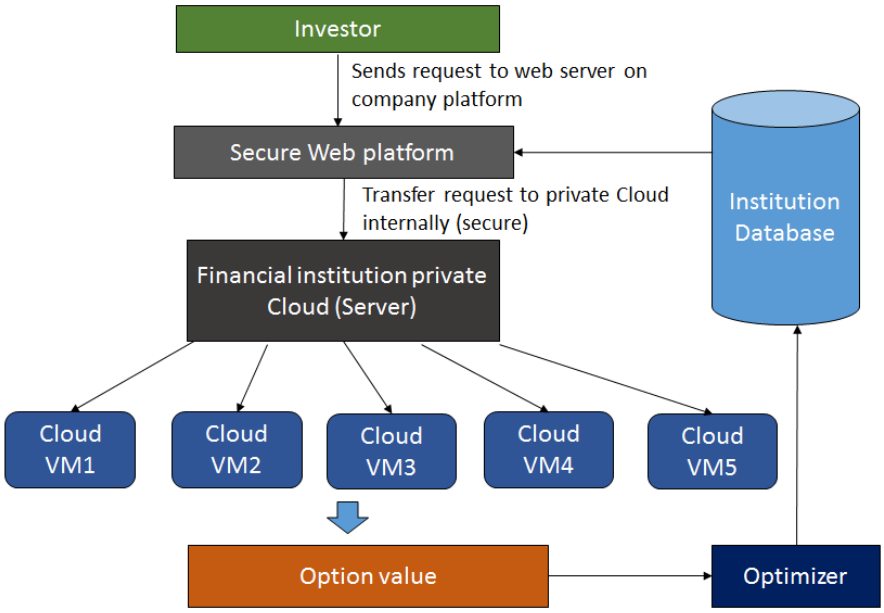



Figure 5.3: SaaS model for application in private Cloud

access the webpage of the financial institution and provide parameters such as stock price, strike price, rate of interest, volatility, time and number of steps then that web page will respond with the option value using the secure channel on the on-premise based private Cloud. The parameters are transferred from secure web page to the Cloud server which can invoke the Cloud VM corresponding to option style (European or American) choice entered by the investor. Hence, the option value can be computed. The same process is triggered multiple times to get an average value with the help of the optimizer for accuracy of the computed option value. Finally, the option value is stored in the database and sent to the investor using the same web page. By using this model, financial institutions can make their resources more secure and agile (easily and rapidly available).

The next subsection discussed performance testing to explore the limitations of

private Clouds for the high performance computing.

## 5.3 Performance Analysis of Cloud VMs using Financial Application

With the application testing using the binomial lattice approach of option pricing, I have shown that the private Cloud can be used to execute a financial application. The next step is to measure the performance of the Cloud VMs in comparison to standalone servers. This can help in analyzing the expected performance that is likely to be obtained through Cloud VMs through comparative analysis between standalone servers and Cloud VMs.

To check the performance of both the systems, I have used Monte-Carlo (MC) simulations for option pricing. MC can test the performance of the system as huge number of simulations are run to obtain value for the option price. To perform this experiment, I have used a Dell R420 standalone server with 12 CPU, 4GB RAM and Ubuntu operating system. For the Cloud VM, the operating system is Ubuntu, RAM is 4GB and the virtual CPUs (vCPU) will be gradually increased with each iteration of the experiment.

For the performance comparison, I have executed MC simulation in a parallel computing environment using MPI (message passing interface) where I can run the parallel tasks for each CPU in the server. I have used Mpich Cluster [25] to create a Cluster of 3 standalone servers to run the MPI code. I created multiple VMs simultaneously with different number of vCPUs. This can produce maximum utilization of

CPUs as they will run the parallel task simultaneously. For the initial test purpose, the standalone server and Cloud VM had 2 CPUs and 2 vCPUs respectively. Operations for MC simulation will run on those 2 CPUs and 2 vCPUs. In the next set of experiments, I gradually increase the number of CPUs and vCPUs in standalone server and Cloud VM to measure the performance with respect to response time to compute results for option value.

I deployed MC code on the VM similar to Binomial lattice application. After the deployment, I ran the experiments on Openstack based Cloud VMs and standalone servers. The performance metric for this experiment by the response time to get the final value of option price from a system.

I ran the experiments multiple times using one configuration for Cloud VM. After averaging the result from multiple experiments, I changed the configuration of VM gradually for each set of experiments. As hypervisor limits Cloud VMs to utilize the resources from more than one node, I suspected how it will perform in case of Cluster configuration. Cluster configuration for this case is when single VM is using resources such as vCPUs of three physical machines. Note that MPI code enables me to use all the CPUs of three physical standalone machines to test the performance of standalone servers.

The parameters for the MC simulation for all the experiments are described in Table 5.7. The input parameters are Stock price, Strike price, Rate of interest, Time, Volatility, Number of Steps and Number of simulations. To utilize the capability of CPUs and vCPUs the number of simulations are kept in the order of 100,000. The input values are kept the same for all the experiments in each iterations.

| Input | Values |
|---|---|
| Stock Price | 50 |
| Strike Price | 40 |
| Rate of Interest | 5% |
| Time | 0.5 year |
| Volatility | 20% |
| Number of Steps | 500 |
| Number of Simulation | 100000 |

Table 5.7: Monte-Carlo Input parameters for experiment

Table 5.8 shows the response time generated after running MC code on the cluster of standalone servers and Cloud VMs. Cloud VMs RAM is kept 4GB for all the iterations.

| Iteration | Server CPU | VM CPUs | VM RAM | Server RT | VM RT |
|---|---|---|---|---|---|
| I-1 | 2 | 2 | 4 | 11799 ms | 9846 ms |
| I-2 | 4 | 4 | 4 | 10539 ms | 8533 ms |
| I-3 | 8 | 8 | 4 | 9155 ms | 7914 ms |
| I-4 | 12 | 12 | 4 | 7717 ms | 7388 ms |

Table 5.8: Performance Analysis using Response Time

As we can see in Table 5.8 and Figure 5.4, as I increased the number of vCPUs for Cloud VM from 2 to 12 it produces response time better than standalone servers cluster. Cloud VMs are more efficent in comparison to standalone server cluster as the time it consumes to distribute processes using MPI on cluster is relatively significant. The overhead of distributing processes is not there in Cloud VM because Cloud has automatic tool for load balancing, hence the Cloud VM performance is better. This led to the question of what if I go across the nodes? I increased the vCPUs beyond 12 for which I have to use other nodes. As discussed earlier, since hypervisor limits the internal configuration of Cloud VM only to a single node of physical machine, I

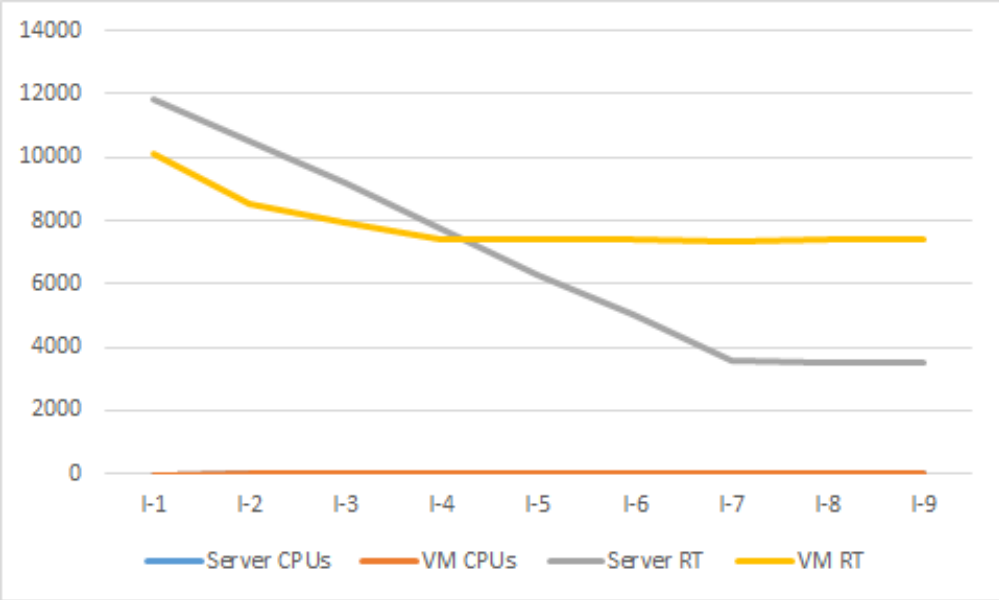



Figure 5.4: Response Time Analysis

suspected that the performance of Cloud VM would be as good as the single node experiments. To test this I did multiple experiments as presented in Table 5.9.

| Iteration | Server CPU | VM CPUs | VM RAM | Server RT | VM RT |
|---|---|---|---|---|---|
| I-1 | 16 | 16 | 4 | 6251 ms | 7389 ms |
| I-2 | 20 | 20 | 4 | 4995 ms | 7385 ms |
| I-3 | 24 | 24 | 4 | 3542 ms | 7371 ms |
| I-4 | 28 | 28 | 4 | 3529 ms | 7375 ms |
| I-5 | 32 | 32 | 4 | 3516 ms | 7381 ms |

Table 5.9: Performance Analysis using Response Time

From Table 5.9, soon after increasing the vCPU from 12, the change in response time seems to have stopped. Response time for Cloud VMs shows that they were unable to utilize the processing power of CPUs as it increased beyond 12 as I suspected. This limitation in performance in this case does not allow Cloud VMs for high speedup, unless some other mechanism are in place to avoid this limitation for

this application.

Another interesting result that can be seen in Table 5.8, 5.9 and Figure 5.4 is that the response time for standalone server cluster becomes almost close to constant beyond 24 CPUs. This could be due to network-bandwidth at some of the processors were used by controller node of my private Cloud. That is, these results show that increasing the number of CPUs can increase the performance of cluster to a certain limit, but we have to consider other factors as well such tasks running on other CPUs for further analysis. This is beyond the scope of my thesis.

# Chapter 6

# Conclusions

To build a private Cloud named Stratus for my thesis I completed several objectives/sub-tasks as described below:

1. To assess and choose the best available open source software for implementing private Cloud: Among various such software I determined the superiority of Openstack before deciding to use it for my private Cloud.

2. To build a private Cloud using the 3-node architecture based on three Dell R420 servers with Intel Xeon processors, 4GB RAM and 500 GB hard disk each. The three nodes are Controller, Compute and Network node as described in Figure 4.5. I implemented Openstack on these three nodes to make them into a small private Cloud.

3. Perform replication testing for VM failure recovery handling and compare replication strategies: With Openstack there are two storage mechanisms Swift and Cinder. I used both to test replication and showed that Cinder is more efficient

than Swift.

In this thesis, I have described how a private Cloud can be built on the Openstack platform, and is used to determine an efficient strategy of volume replication using Cinder. Volume replication provides an efficient storage mechanism in comparison to object based replication using Swift. The presented approach represents an effective replication solution for Openstack private Clouds. With the mechanism proposed, I can implement similar strategies in different private Cloud platforms such as Cloudstack, OpenNebula and Eucalyptus.

There are some limitations in public Clouds as their data-centers can be situated in different geo-locations which can add to hop count wait time and produce delay in replication In such scenarios a block-based strategy such as Cinder is clearly preferable. In my particular environment with a relatively small dataset, Cinder was proven to be the preferable choice as it had a significant advantage in terms of the time it takes to create a backup and deploy it. However, for data consistency either Cinder or Swift could be used since my results showed no data loss post deployment of VMs using either storage mechanism..

4. Perform application testing using option pricing problem of finance: Investment being a data private operation, I decided to develop and test an option pricing financial applications on my private Cloud. I successfully ran one of the option pricing approaches (Binomial lattice) on the private Cloud. I showed that the option values computed are accurate compared to a closed-form solution given by Black-Scholes-Merton model.

   With the application testing, I distributed the European and American style of

options on various VMs to generate the value of option price using single server which communicated with other VMs. This distributed system within a private Cloud functioned correctly for small scale computations.

5. Performance testing on the basis of response time and compare with standalone servers: I also evaluated performance of the private Cloud using the response time feature of Linux and MC simulation code. It produced some interesting results but furthermore, it helped me understand how standalone server cluster can be used for high performance computing, if required, in the current setup. It also helped me to understand the role of hypervisor in the virtualization environment. Hypervisor limits the Cloud VM to remain in a single node. In order to make use of the Cloud VM from multiple nodes, we need to have a distributed file system and inter node communication as well.

My thesis research included reasonably extensive experimental work with the private Cloud. Use of finance application added a new dimension to my work in respect to learning and implementation. During the literature review, implementation and experimentation, my knowledge in the field of private Cloud and computational finance was enhanced significantly.

## Future Work

Many other applications can be deployed on private Clouds and use features. For fields such as finance, military and hospitals where data security and easy data accessibility are a major concern, private Clouds can play a big role to fulfill their re-

quirements. My experiments with option pricing technique of binomial lattice showed that Cloud VMs can perform in an effective way and generate accurate results for cloud-based applications. My option pricing application also provides a prototype model for financial institutions for their internal and external operational use.

I also proposed a model for SaaS application environment in a financial institution. This model helps the institution to achieve two major objectives: Security and agile data availability.

To make Cloud system work as a cluster, distributed file system such as Hadoop (HDFS) is required. I have not explored features of HDFS like systems but it might produce captivating results with use of Openstack, which I leave as future work.

# Appendix A

# Openstack implementation

**Note:** The following major steps are prescribed by Openstack document [2]. Hence, some of the description are from the Openstack document such as Figure A.2.

All three servers are of high performance capability and operability. Below are some major implementation steps followed during deployment of private Cloud using openstack:

1. Configure Ubuntu server 14.04 on the controller and compute nodes for the primary OS installation. Install RabbitMQ message broker service and MYSQL during the time of OS installation.

2. Setup the IP address for all the nodes as shown in Figure A.1.

3. Configure DNS server on the controller node for the name resolution. Editing the host configuration file to identify the node as controller, compute and network node.

4. Configure DHCP on the controller node to obtain IP address for the virtual

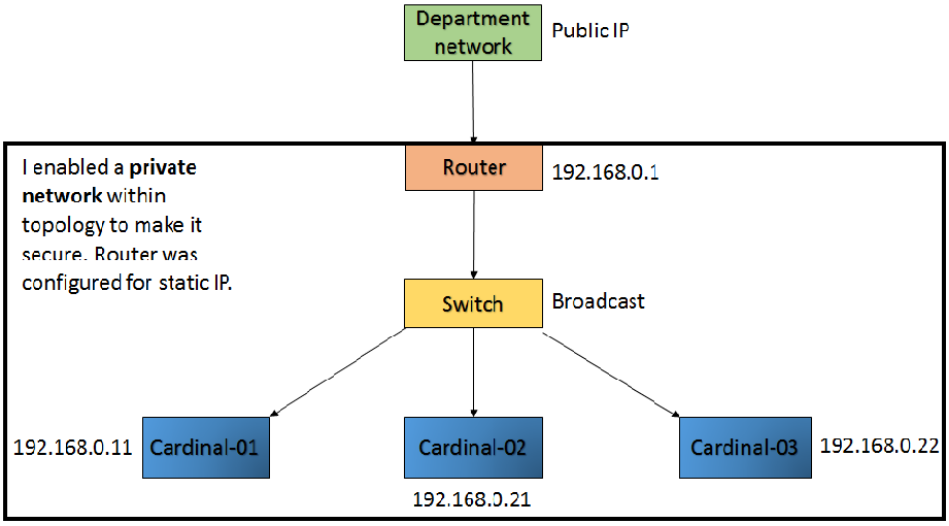



Figure A.1: Network configuration Diagram

machines that would be configured later.

5. Configure Network time protocol (NTP) server at a controller node that will propagate the time settings to the connected nodes through the address resolution protocol.

6. Install Keystone services (Identity Service) on the controller node for the authorization. Create following elements (shown in Figure A.2) within the Keystone service by using a MYSql database.

7. These elements are created with some key values. Below is the screenshot for user-list (Figure A.3), service-list (Figure A.4), tenant-list (Figure A.5) and user-role list (Figure A.6).

   Figure A.3 shows the list of users which communicate with the keystone for authorization before any operation. Once they authorize using the get-token

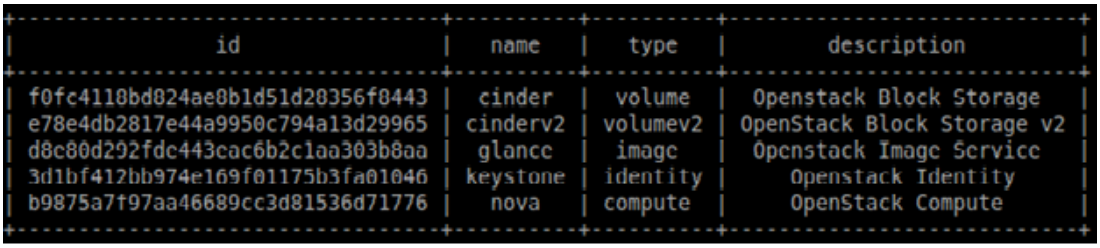
| Elements | Description |
|---|---|
| User | Digital representation of a person, system, or service who uses OpenStack cloud services. |
| Credentials | Data that is known only by a user that proves who they are. |
| Authentication | The act of confirming the identity of a user. |
| Token | An arbitrary bit of text that is used to access resources. |
| Tenant | A container used to group or isolate resources and/or identity objects. |
| Service | Provides one or more endpoints through which users can access resources and perform operations. |
| Endpoint | A network-accessible address, usually described by a URL, from where you access a service |
| Role | A personality that a user assumes that enables them to perform a specific set of operations |

Figure A.2: Elements of Identity service

function of keystone and using secret-id, they are ready to perform any operations. Figure A.4 shows the list of services and their description, these services have particular id and they operate on the basis of token authentication. Figure A.5 shows the tenant list which resides on the users for the access to the services. Figure A.6 shows the user role list where administrative and member access are defined, these have the privileges to perform user defined operations.

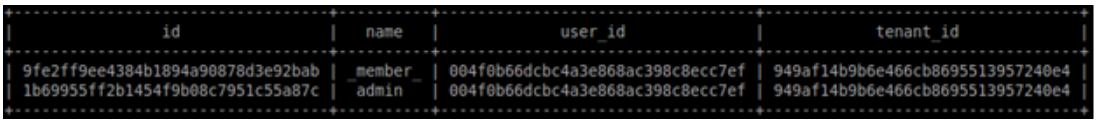
```
+----------------------------------+--------+---------+--------------------------+
|                id                |  name  | enabled |          email           |
+----------------------------------+--------+---------+--------------------------+
| 004f0b66dcbc4a3e868ac398c8ecc7ef | admin  |   True  | bajpaideepak81@gmail.com |
| ab0de13303a54215ac513ad053455e5d | admin1 |   True  |  bajpaid@myumanitoba.ca  |
| 7a5e6701241f485c9d026b0007bf4e80 | cinder |   True  | bajpaideepak81@gmail.com |
| 4e2fae198d2848cab50a6c7ba3214bd2 |  demo  |   True  | bajpaid@cs.umanitoba.ca  |
| 539ec06d17d3499dab202c71a7318c84 | glance |   True  | bajpaideepak81@gmail.com |
| 46d6c0737fa844dc8d372a8795aec44f |  nova  |   True  | bajpaideepak81@gmail.com |
+----------------------------------+--------+---------+--------------------------+
```

Figure A.3: Keystone user-list



| id | name | type | description |
|---|---|---|---|
| f0fc4110bd824ae8b1d51d20356f8443 | cinder | volume | Openstack Block Storage |
| e78e4db2817e44a9950c794a13d29965 | cinderv2 | volumev2 | OpenStack Block Storage v2 |
| d8c80d292fdc443cac6b2c1aa303b8aa | glancc | imagc | Opcnstack Imagc Scrvicc |
| 3d1bf412bb974e169f01175b3fa01046 | keystone | identity | Openstack Identity |
| b9875a7f97aa46689cc3d81536d71776 | nova | compute | OpenStack Compute |

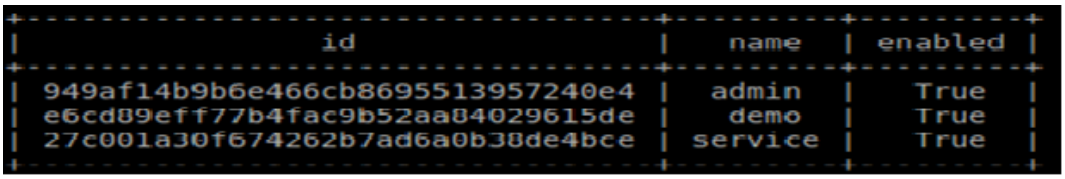
Figure A.4: Keystone service-list



| id | name | user_id | tenant_id |
|---|---|---|---|
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ | 004f0b66dcbc4a3e868ac398c8ecc7ef | 949af14b9b6e466cb8695513957240e4 |
| 1b69955ff2b1454f9b08c7951c55a87c | admin | 004f0b66dcbc4a3e868ac398c8ecc7ef | 949af14b9b6e466cb8695513957240e4 |

Figure A.5: Keystone tenant-list

8. Once the users are created for the Keystone authentication, Glance needs to be configured. Glance retrieves the VM OS images that can be used by nova compute for provisioning. It uses MYSQL for database entry and store all the image in swift in the form of an object. These object images can be extracted whenever requested. Glance can manually or automatically download the image from the image provider. As we can see in Figure A.7, images are given an ID and disk format, they are kept in active state if mounted on glance for provisioning, but if it is stored in swift then it is in passive state. Figure A.7 shows the sample VM bootable images that can be used as per the requirement.



| id | name | enabled |
|---|---|---|
| 949af14b9b6e466cb86955139572 40e4 | admin | True |
| e6cd89eff77b4fac9b52aa84029615de | demo | True |
| 27c001a30f674262b7ad6a0b38de4bce | service | True |

Figure A.6: Keystone user-role list


Figure A.7: VM bootable image

9. As three node architecture is used here, the nova legacy network is used where network node is integrated within a controller node. In this case, dhcp agent is installed on the compute nodes. First, network api is setup on the controller node to communicate compute node so that floating IP address can be provided by the controller node to the VM on the compute node. Secondly, the network services are configured on the compute nodes stating the network interface and other details to communicate with controller node.

10. Nova is installed on the controller and compute node with the metadata file on the compute and api, conductor, scheduler, certificate, novncproxy and consoleauth file on the controller node. All these components have their role in the VM provisioning as scheduler schedule the tasks on the VM and novncproxy helps in remote access to the instance. Nova-Api helps in communication and certificate authorize the services. They collectively act as a heart of this system.

11. Once we have an image, provision service and network, then space is require for the data and containers which is provided by the cinder component. Cinder is installed on the controller or can be configured as a separate as cinder node. Cinder provides block storage for the space of the VM and backup of data. Block storage enhances the speed of the data access as it provided high IOPs value. Volumes are created on the compute node which can later use for the

space provisioning for the VM. Figure A.8 shows the volume on the cinder.

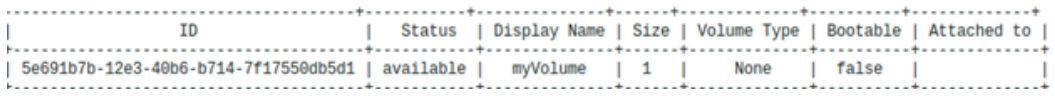



Figure A.8: Cinder volume

12. The rest of the services are optional and can be configured if required by the user, but I limited to the above mentioned services for this project. While configuring all these services, endpoints are created so the components have the knowledge to connect to url address to resolve the request. Figure A.9 shows the endpoints for the above mentioned components.


Figure A.9: Service-List endpoints

## A.1 Major Issues during implementation and their Solutions

Below are some major issues that I faced while implementing components of Openstack. There were some minor issues as well, but those were resolved with the help of hit and trial and easy configuration changes. Major issues are:

1. Keystone authentication requires OS URL authorization and OS service point authorization for the further implementation. To overcome this issue, a shell file was created with administrator privileges containing below

   export OS USERNAME is admin

   export OS PASSWORD is ADMIN PASS

   export OS TENANT NAME is admin

   export OS AUTH URL is http://controller:35357/v2.0

   This file was sourced through root, so after reboot it is automatically considered by the Openstack environment. But the issue persists and to get an automatic OS username and OS password on each authentication file openrc was created to source the username and password for the admin control. This permanent authentication helps to avoid multilevel authorization on each step.

2. Glance authentication with Keystone requires glance-api.conf file with tokens and other configuration description so that it can communicate with the database and keystone. It also requires rabbitmq for message passing. The default userid and password for rabbitmq have to be extracted or changed to input into configuration file so that it can communicate with the message passing protocol for message passing. All services need to be restarted for further implementation.

3. Oslo.config has to be rebuilt to obtain image retrieval from Glance or it will throw an exception error as import files missing. Using pip installer oslo.config have to be reinstalled with the new version and import file have to be uncomment out in glance-api.conf.

4. In Nova controller, port 8774 was not listening and unable to communicate to keystone and rabbitmq to allow services to resume. To open the port nova-api.config has to be edited with the correct configuration details.

5. Cinder volumes was in creating status and bootable have to be defined in the cinder.conf for the endpoint status. The module was not functional until the local.conf was enabled with the correct configuration. Block memory address resides in the cinder.api library which in response contact controller node for keystone authorization. To setup this configuration I made an ethernet port of controller node accessible to compute node.

## A.2 Configuration for Openstack nodes

**Configuration for Controller node**

```
[[local|localrc]]
HOST_IP=192.168.0.106
FLAT_INTERFACE=em1
FIXED_RANGE=10.4.128.0/20
FIXED_NETWORK_SIZE=4096
FLOATING_RANGE=192.168.0.128/25
MULTI_HOST=1
LOGFILE=/opt/stack/logs/stack.sh.log
ADMIN_PASSWORD=*secret*
DATABASE_PASSWORD=root
```

```
RABBIT_PASSWORD=root

SERVICE_PASSWORD=root

SERVICE_TOKEN=xyzpdqlazydog

#Enable celiometer

enable_plugin ceilometer git://git.openstack.org/openstack/ceilometer

#Enable sahara

enable_plugin sahara git://git.openstack.org/openstack/sahara

# Enable sahara-dashboard

enable_plugin sahara-dashboard

git://git.openstack.org/openstack/sahara-dashboard
```

### Configuration for Compute node

```
[[local|localrc]]

HOST_IP=192.168.0.103 # change this per compute node

FLAT_INTERFACE=em1

FIXED_RANGE=10.4.128.0/20

FIXED_NETWORK_SIZE=4096

FLOATING_RANGE=192.168.0.128/25

MULTI_HOST=1

LOGFILE=/opt/stack/logs/stack.sh.log

ADMIN_PASSWORD=*secret*

DATABASE_PASSWORD=root

RABBIT_PASSWORD=root

SERVICE_PASSWORD=root
```

```
SERVICE_TOKEN=xyzpdqlazy

DATABASE_TYPE=mysql

SERVICE_HOST=192.168.0.101

MYSQL_HOST=$SERVICE_HOST

RABBIT_HOST=$SERVICE_HOST

GLANCE_HOSTPORT=$SERVICE_HOST:9292

ENABLED_SERVICES=n-cpu,n-net,n-api-meta,c-vol

NOVA_VNC_ENABLED=True

NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_auto.html"

VNCSERVER_LISTEN=$HOST_IP

VNCSERVER_PROXYCLIENT_ADDRESS=$VNCSERVER_LISTEN

#Enable celiometer

enable_plugin ceilometer git://git.openstack.org/openstack/ceilometer

#Enable sahara

enable_plugin sahara git://git.openstack.org/openstack/sahara

# Enable sahara-dashboard

enable_plugin sahara-dashboard

git://git.openstack.org/openstack/sahara-dashboard
```

## A.3 Shell scripts used in Replication Testing

1. Scripts: To generate text files with n size of data

generate_data.sh:

```
for i in {1..n}; do echo "Hello World $i" > "datafile"$i".txt";
done
for i in {1..n}; do for j in {1..19};
do cat "datafile"$i".txt" "datafile"$i".txt" >
test.txt && mv test.txt
"datafile"$i".txt"; done; done
```

2. Scripts: These were the scripts used to generate the VM snapshots every 2 hours.

testing_run_scripts.sh:

```
#!/bin/bash
IMAGENUM=1
TARGET="server1"
for i in 1 2 3 4; do
sh freeze_instance.sh & sh make_snapshot.sh $IMAGENUM $TARGET &
IMAGENUM=$(( (IMAGENUM + 1)%5 ))
sleep 2h
done
echo Script Complete...
```

freeze_instance.sh:

```
#!/bin/bash
```

```
sleep 2m

sync

echo "freezing..."

ssh -i my-private-key.txt ubuntu@10.0.0.4

sudo mount -a;

sudo fsfreeze -f /mnt;

sleep 3m;

sudo fsfreeze -u /mnt'

echo "unfreezing..."

echo "DONE FREEZE_INSTANCE.SH"
```

make_snapshot.sh:

```
#!/bin/bash

echo "deleting..."

nova image-delete $2"_snapshot$1"

sleep 2m

echo "making snapshot..."

t1=$(date +"%s")

nova image-create --poll $2 $2"_snapshot$1"

t2=$(date +"%s")

diff=$(expr $t2  $t1)

echo "Number of seconds to create snapshot"+$2+"_snapshot$1 :
```

```
$diff"

echo "DONE INSTANCE_SNAPSHOT.SH"
```

3. Scripts: These scripts were used to create the volume snapshots

testing_volume_scripts.sh:

```
#!/bin/bash

IMAGENUM=1

TARGET="cloudvolume"

for i in 1 2 3 4; do

sh volume_snapshot.sh $IMAGENUM $TARGET &

IMAGENUM=$(( (IMAGENUM + 1)%5 ))

sleep 2h

done

echo Script Complete...
```

volume_snapshot.sh:

```
#!/bin/bash

echo "deleting volume..."

nova volume-snapshot-delete $2"-snapshot$1"

sleep 3s

t1=$(date +"%s")

echo "$t1"

cinder snapshot-create --force
```

```
True --display-name $2"-snapshot$1" $2 & EPID=$!

wait $EPID

t2=$(date +"%s")

echo "$t2"

diff=$(expr $t2 - $t1)

echo "Number of seconds to create volume snapshot : $diff"
```

# Bibliography

[1] Gartner disaster recovery. http://www.gartner.com/it-glossary/dr-disaster-recovery.html/. Online: Accessed; 2016-07-02.

[2] Openstack installation document. http://docs.openstack.org/juno/install-guide/install/apt/content/. online: Accessed; 2015-01.

[3] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Shiraz, Abdullah Yousafzai, and Feng Xia. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications*, 52:11–25, 2015.

[4] Deepak Bajpai and Ruppa K. Thulasiram. Comparing replication strategies for financial data on openstack based private cloud. In *Proc. of the IARI International Conf. on Cloud Computing, 2016, Rome, Italy*, pages 139–144. IARIA, 2016.

[5] Béatrix Barafort, Bernard Di Renzo, and Olivier Merlan. Benefits resulting from the combined use of iso/iec 15504 with the information technology infrastructure library (itil). In *Product Focused Software Process Improvement*, pages 314–325. Springer, 2002.

[6] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The journal of political economy*, pages 637–654, 1973.

[7] Phelim Boyle, Mark Broadie, and Paul Glasserman. Monte-carlo methods for security pricing. *Journal of economic dynamics and control*, 21(8):1267–1321, 1997.

[8] Phelim P Boyle. Options: A Monte-Carlo approach. *Journal of financial economics*, 4(3):323–338, 1977.

[9] Rajkumar Buyya, Christian Vecchiola, and S Thamarai Selvi. *Mastering cloud computing: foundations and applications programming*. Newnes, 2013.

[10] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *10th IEEE International Conference on High Performance Computing and Communications, 2008. HPCC'08.*, pages 5–13.

[11] Bin Cai, Feng Xu, Feng Ye, and Wenhuan Zhou. Research and application of migrating legacy systems to the private cloud platform with cloudstack. In *Automation and Logistics (ICAL), 2012 IEEE International Conference on*, pages 400–404.

[12] Victor Chang. Towards a big data system disaster recovery in a private cloud. *Ad Hoc Networks*, 35:65–82, 2015.

[13] Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for I/O pro-

cessing in the xen virtual machine monitor. In *USENIX Annual Technical Conference, General Track*, volume 50, 2005.

[14] Antonio Corradi, Mario Fanelli, and Luca Foschini. VM consolidation: A real case based on openstack cloud. *Future Generation Computer Systems*, 32:118–127, 2014.

[15] John C Cox, Stephen A Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.

[16] Barb Darrow. Amazon Web Services threatens all IT incumbents. https://gigaom.com/2013/05/29/killer-cloud-report-says-amazon-web-services-threatens-all-it-incumbents/, 2013. [Online; accessed 29-May-2015].

[17] Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, Everett Toews, and Joe Topjian. *OpenStack Operations Guide.* " O'Reilly Media, Inc.", 2014.

[18] Paul Glasserman and Bin Yu. Simulation for american options: regression now or regression later? In *Monte-Carlo and Quasi Monte-Carlo Methods 2002*, pages 213–226. Springer, 2004.

[19] Sumit Goyal. Public vs private vs hybrid vs community-cloud computing: A critical review. *International Journal of Computer Network and Information Security (IJCNIS)*, 6(3):20, 2014.

[20] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in cloud computing: What it is, and what it is not. In *ICAC*, pages 23–27, 2013.

[21] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*, pages 89–96. IEEE.

[22] John Hull and Alan White. The pricing of options on assets with stochastic volatilities. *The journal of finance*, 42(2):281–300, 1987.

[23] John C Hull. *Options, futures, and other derivatives*. Pearson Education India, 2006.

[24] Xiaoen Ju, Livio Soares, Kang G Shin, Kyung Dong Ryu, and Dilma Da Silva. On fault resilience of openstack. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 2. ACM, 2013.

[25] Guillaume Mercier and Olivier Aumage. Mpich/madiii: a cluster of clusters-enabled mpi implementation. In *Third IEEE International Symposium on Cluster Computing and the Grid (CCGRID'03)*, pages 26–35, 2003.

[26] Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144, 1976.

[27] Stanislav Milanovic and Zoran Petrovic. Building the enterprise-wide storage area network. In *EUROCON'2001, Trends in Communications, International Conference on.*, volume 1, pages 136–139. IEEE.

[28] Dejan Milojičić, Ignacio M Llorente, and Ruben S Montero. OpenNebula: A cloud management tool. *IEEE Internet Computing*, (2):11–14, 2011.

[29] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131.

[30] Vinit Padhye and Anand Tripathi. Scalable transaction management with snapshot isolation on cloud data management systems. In *Cloud computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 542–549.

[31] Ken Pepple. *Deploying openstack*. " O'Reilly Media, Inc.", 2011.

[32] S Rakhmayil, I Shiller, and RK Thulasiram. Different estimators of the underlying asset's volatility and option pricing errors: Parallel Monte-Carlo simulation. In *Intl. Conf. on Computational Finance and its Applications (ICCFA), April 21-23, 2004, Bologna, Italy.*, pages 121–131.

[33] Rajiv Ranjan, Rajkumar Buyya, Surya Nepal, and Dimitrios Georgakopulos. A note on resource orchestration for cloud computing. *Concurrency and Computation: Practice and Experience*, 27(9):2370–2372, 2015.

[34] Bahman Rashidi, Mohsen Sharifi, and Talieh Jafari. A survey on interoperability in the cloud computing environments. *International Journal of Modern Education and Computer Science (IJMECS)*, 5(6):17, 2013.

[35] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 44–51. Ieee, 2009.

[36] Ranjan Saha, Bhanu Sharma, Ruppa K Thulasiram, and Parimala Thulasiraman. A novel architecture for financial investment services on a private cloud. In *Algorithms and Architectures for Parallel Processing*, pages 370–379. Springer, 2013.

[37] Eduardo S Schwartz and Walter N Torous. Prepayment and the valuation of mortgage-backed securities. *The Journal of Finance*, 44(2):375–392, 1989.

[38] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3), 2012.

[39] Borja Sotomayor, Rubén S Montero, Ignacio M Llorente, and Ian Foster. Virtual infrastructure management in private and hybrid clouds. *Internet computing, IEEE*, 13(5):14–22, 2009.

[40] Satish Srirama, Oleg Batrashev, and Eero Vainikko. Scicloud: scientific computing on the cloud. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 579–580.

[41] George Suciu, Elena G Ularu, and Razvan Craciunescu. Public versus private cloud adoption  a case study based on open source cloud platforms. In *20th Telecommunications Forum (TELFOR), 2012*, pages 494–497. IEEE.

[42] Xiaolong Wen, Genqiang Gu, Qingchun Li, Yun Gao, and Xuejie Zhang. Comparison of open-source cloud management platforms: Openstack and openNeb-

ula. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2457–2461. IEEE.