



**Tribhuvan University**  
**Institute of Science and Technology**

**Automated Live Migration of Virtual Machines in Cloud Data Center**

**A Dissertation**

Submitted To

**Central Department of Computer Science & Information Technology**

Tribhuvan University,  
Kirtipur, Kathmandu, Nepal

In Partial Fulfillment of the Requirements for the Master's Degree of Science

in

Computer Science & Information Technology

By:

**Ramesh Tamang**  
**(15<sup>th</sup> September 2019)**



**Tribhuvan University**  
**Institute of Science and Technology**

**Automated Live Migration of Virtual Machines in Cloud Data Center**

**A Dissertation**

Submitted To

**Central Department of Computer Science & Information Technology**

Tribhuvan University, Kirtipur, Kathmandu, Nepal

In Partial Fulfillment of the Requirements for the

**Master's Degree in Computer Science & Information Technology**

By:

**Ramesh Tamang**

**(15<sup>th</sup> September 2019)**

Supervisor

**Prof. Dr. Shashidhar Ram Joshi (IOE, TU)**



**Tribhuvan University**  
**Institute of Science and Technology**  
**Central Department of Computer Science & Information Technology**

**Student's Declaration**

I hereby declare that I am the only author of this work and no sources other than the listed here have been used in this work.

.....

Ramesh Tamang

Date: 15<sup>th</sup> September 2019



**Tribhuvan University**  
**Institute of Science and Technology**  
**Central Department of Computer Science & Information Technology**

**Supervisor's Recommendation**

I hereby recommend that this dissertation prepared under my supervision by Mr. Ramesh Tamang entitled “**Automated Live Migration of Virtual Machines in Cloud Data Center**” in partial fulfilment of the requirements for the degree of M.Sc. in Computer Science and Information Technology.

---

**Prof. Dr. Shashidhar Ram Joshi**

Department of Electronics and Computer Engineering

Institute of Engineering,

Pulchowk, Lalitpur, Nepal.

Date: 15<sup>th</sup> September 2019



**Tribhuvan University**  
**Institute of Science and Technology**  
**Central Department of Computer Science & Information Technology**

**LETTER OF APPROVAL**

We certify that we have read this dissertation work and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfilment for the requirement of Master's Degree in Computer Science and Information Technology.

**Evaluation Committee**

---

**Asst. Prof. Nawaraj Paudel**

Central Department of Computer Science  
& Information Technology,  
Tribhuvan University, Kathmandu, Nepal  
**(Head of Department)**

---

**Prof. Dr. Shashidhar Ram Joshi**

Department of Electronics & Computer  
Engineering, Institute of Engineering,  
Pulchowk, Kathmandu, Nepal  
**(Supervisor)**

---

Mr. Lochan Lal Amatya  
**(External Examiner)**

---

Mr. Bal Krishna Subedi  
**(Internal Examiner)**

**Date: 15<sup>th</sup> September 2019**

## Acknowledgement

Firstly, I would like to express my sincere gratitude to my respected supervisor **Prof. Dr. Shashidhar Ram Joshi** of Institute of Engineering (IOE) for his generous advice, inspiring guidance and encouragement throughout my research work. His guidance helped me in all the time of research and writing of this dissertation. Without his kind and patient review of this work, it is impossible to complete my research.

Beside my supervisor, I would also like to thank **Mr. Jagdish Bhatta**, faculty member of Central Department of Computer Science and Information Technology (CDCSIT) for his insightful comments and valuable suggestions.

Similarly, I would like to extend my gratitude to **Asst. Prof. Nawaraj Paudel (HOD)** CDCSIT, faculties Prof. Dr. Subarna Shakya, Mr. Dheeraj Kedar Pandey, Mr. Sarbin Sayami, Mr. Bikash Balami, Mr Arjun Saud and Mrs. Lalita Sthapit of Central Department of Computer Science and Information Technology, Tribhuvan University.

Last but not the least, I would like to thank my parents and to my sister Anju Tamang for supporting me spiritually throughout writing this dissertation and my life in general.

## **Abstract**

The thesis entitled “**Automated Live Migration of Virtual Machines in Cloud Data Center**” involves the study and implementation of load balancing algorithm for Virtual Machine Management in the cloud infrastructure. The study is focused on dynamic resource scaling and live migration algorithm of virtual machines to achieve the load balancing in the cloud infrastructure. As data centers grow sharply, they find themselves accommodating an increasing number of Physical Machines and Virtual Machines. This development requires an effective resource management in data centers. As a result, the load is evenly distributed and service level agreements (SLAs) are met. From this point of view, load balancing and live migration are becoming essential processes for data center management. In cloud, computing resources are provided as a service to its clients across the globe based on demand. Huge demand for cloud resources results in overutilization by degrading the performance of the servers whenever there is a heavy load. It is necessary to distribute the load across the servers in cloud by taking into consideration of allocating the right amount of resources dynamically based on the load to improve the performance of applications running inside virtual machines. Experiment is conducted on Xen servers in the real physical hardware and in the virtualized environment and implemented dynamic resource scaling and live migration technique to manage load in the virtual machines running on those Xen servers. The response time of virtual machine is used as a metric to measure the performance of algorithms. After the implementation of dynamic resource scaling and live migration technique, the improved performance of virtual machine in terms of response time and efficient utilization of Xen server resources is observed.

### **Keywords:**

*Cloud Computing, Virtualization, Virtual Machine, Hypervisor, Automated Live Migration, XenServer, Credit Scheduler, Resource Scaling, Load Balancing*

To my dear family



# Table of Contents

Acknowledgment	i
Abstract	ii
List of Figures	viii
List of Tables	x
Abbreviations	xi
<b>CHAPTER 1 INTRODUCTION AND BACKGROUND</b>	<b>1</b>
1.1 Cloud Computing	1
1.2 Virtualization	1
1.3 Virtual Machine Migration	2
1.3.1 Off-line Migration	2
1.3.2 Live Migration	2
1.4 Automating Live Migration Process	4
1.5 Load Balancing	4
1.6 Problem Definition	5
1.7 Objectives	6
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>7</b>
<b>CHAPTER 3 METHODOLOGY</b>	<b>8</b>
3.1 Xen	8
3.2 CPU Scheduler in Xen	8
3.2.1 Borrowed Virtual Time	8

3.2.2 Simple Earliest Deadline First	8
3.2.3 Credit Scheduler	8
3.3 System Architecture	9
3.3.1 Resource Scaling Down Activity	10
3.3.2 Resource Scaling Up Activity	11
3.3.3 Live Migration Activity	12
3.4 Algorithm	12
3.4.1 Load Balancing and Live Migration Algorithm	12
3.4.2 Algorithm for Scaling	15
<b>CHAPTER 4 IMPLEMENTATION</b>	<b>17</b>
4.1 Experimental Setup	17
4.2 System Specifications	19
4.3 Oracle VM VirtualBox	21
4.4 XenServer	21
4.4.1 Introduction to XenServer	21
4.4.2 Installing XenServer in Physical Hardware and Virtual Box	22
4.5 Citrix XenCenter	26
4.5.1 Introduction to XenCenter	26
4.5.2 Installation	27
4.6 Network File System	27
4.6.1 Introduction to NFS	27

4.6.2 Installation	27
4.7 VM install and Management in XenServer	28
4.7.1 Using XenCenter GUI	28
4.7.2 Using CLI	34
4.7.3 XenServer and VM Management Commands in XenServer	35
4.8 Apache	36
4.9 Httpperf	37
4.10 Stress	38
4.11 Implementation of Algorithms	38
4.11.1 Shell Script	38
4.11.2 Functions	39
<b>CHAPTER 5 EXPERIMENT AND OBSERVATION</b>	<b>40</b>
5.1 Response Time in One-Minute Load Average from Physical Test Environment	41
5.2 Response Time in Five-Minute Load Average from Physical Test Environment	42
5.3 Response Time in One-Minute Load Average from Virtualized Test Environment	43
5.4 Response Time in Five-Minute Load Average from Virtualized Test Environment	44
5.5 Total Migration Time Calculation	44
5.6 Evaluation	45
<b>CHAPTER 6 CONCLUSION AND FUTURE WORK</b>	<b>47</b>
6.1 Conclusion	47
6.2 Future Work	47

References	48
Appendix A Screen Shots	50
Appendix B Source Code	58

## List of Figures

Figure 3.1: System Architecture.....	10
Figure 3.2: Sequence of activities to scale down the resources.....	11
Figure 3.3: Sequence of activities to migrate VM to suitable target server.....	12
Figure 4.1: Experimental set up of XenServer cloud platform in physical hardware.....	17
Figure 4.2: Experimental set up of XenServer cloud platform in virtual environment.....	18
Figure 4.3: XenServer Installation Start Console.....	22
Figure 4.4: XenServer Installation Progress Bar.....	23
Figure 4.5: XenServer Installation Completion.....	24
Figure 4.6: XenServer System Configuration Console.....	25
Figure 4.7: Xen Server Login Console.....	26
Figure 4.8: XenCenter Download from Xenserver.....	27
Figure 4.9: Start Citrix XenCenter software.....	29
Figure 4.10: XenCenter VM Management Menu.....	29
Figure 4.11: VM Selection Template.....	30
Figure 4.12: Enter the name of the operating system.....	30
Figure 4.13: Selection of installation media.....	31
Figure 4.14: CPU and Memory allocation.....	32
Figure 4.15: Disk information for VM.....	32
Figure 4.16: Configure Network Settings.....	33
Figure 4.17: VM Console in XenCenter.....	34
Figure 5.1 Response Time in One-Minute Load Average from Physical Test Environment.....	41

Figure 5.2 Response Time in Five-Minute Load Average from Physical Test Environment.....42

Figure 5.3 Response Time in One-Minute Load Average from Virtualized Test Environment....43

Figure 5.4 Response Time in Five-Minute Load Average from Virtualized Test Environment...44

## **List of Tables**

Table 4.1: Resource Allocation for XenServer Platform Setup on dedicated hardware.....	20
Table 4.2: Resource Allocation for XenServer Platform in VirtualBox.....	20
Table 5.1: Total Migration Time Calculation of VM live migration in Physical Test Environment....	45
Table 5.2: Total Migration Time Calculation of VM live migration in Virtual Test Environment.....	45

## **Abbreviations**

CLI	Command Line Interface
CSP	Cloud Service Provider
DNS	Domain Name System
GUI	Graphical User Interface
IAAS	Infrastructure as a Service
IP	Internet Protocol
NFS	Network File System
NUC	Next Unit of Computing
PAAS	Platform as a Service
SAAS	Software as a Service
SLA	Service Level Agreement
VM	Virtual Machine



# **CHAPTER 1**

## **INTRODUCTION AND BACKGROUND**

### **1.1 Cloud Computing**

Cloud Computing is a new paradigm where computing resources are made available to the users on demand in the pay as you go model. Cloud Computing provides computing resources mainly in three forms which are: Software as a Service (SaaS), Platform as Service (PaaS) and Infrastructure as a Service (IaaS). In SaaS, a Software application is made available to the users. One of the examples of SaaS is Salesforce. In PaaS, application development platform is made available to the users. Microsoft Azure is one of the examples for PaaS. In IaaS, an Infrastructure is provided as a service to the users. Amazon, Rackspace are some of the providers who offer IaaS service to the users.

Cloud based services play a major role in collaborative computing on the Internet across the globe. Managing the computing and network resources to serve dynamic demands of the tenants of an infrastructure-as-a-service (IaaS) cloud is a major challenge for the cloud service provider (CSP). Accommodating the custom resource requirements often result in the migration of virtual machines (VMs) already executing on a server. Live VM migrations, which relocates a VM without halting, helps the CSP to achieve this objective [1].

The essential characteristics of cloud computing are rapid elasticity, on-demand self-service, resource pooling, broad network access, and measured service [2]. Cloud Computing Deployment Model refers to the location and management of the infrastructure cloud services. The deployment models of cloud computing are Private Cloud, Community Cloud, Public Cloud and Hybrid Cloud. The key technology behind cloud computing is virtualization and it provides a means for migration of virtual machines from one server to another.

### **1.2 Virtualization**

Virtualization in cloud computing is the ability to run multiple operating systems on a single physical system and share the underlying hardware resources. Some of the benefits of virtualization are Server Consolidation, Resource Load Balancing, Backup and Restore, Security and Scaling. The way a virtualized server functions is that it introduces an abstraction layer

between the hardware and the operating system. This abstraction layer is called the hypervisor [2]. The main characteristics of virtual machine are isolation, ease of testing and mobility.

Hypervisor are divided into two types which are Bare Metal Hypervisor and Hosted Hypervisor. The Bare Metal Hypervisor is installed on the hardware itself which has a significantly higher level of control over the resources available to it because it does not have to go through an intermediary to access the resources. Some of the examples of bare metal hypervisor are Xen, VMware and Hyper-V. The Hosted Hypervisor is not installed on the hardware itself, but within an operating system on the server, and thus it is under the control of that operating system. Some of the example of Hosted Hypervisor are VMware Player, VirtualBox.

### **1.3 Virtual Machine Migration**

Virtual machine migration is the process of moving virtual machine from one physical host to another physical host with minimal impact on the operation of other virtual machines. The virtual machine migration includes migration of entire operating system and all of its applications as one unit between physical machines. The migration of VM can be divided into two categories which are Off-line VM Migration and Live VM Migration.

#### **1.3.1 Off-line Migration**

Off-line migration is also called Non-Live Migration where VM is paused at the source host and all the states of VM is transferred to destination host and then finally resuming of VM is done at the new physical host. The advantage of offline migration is that the process is simple, but the disadvantage is larger down time.

#### **1.3.2 Live Migration**

Live Migration is a key feature of virtualization. It is the process of moving the VM from one physical host to another without interrupting any of the VM running services. The key advantage of live migration is user-invisible down time with fast network.

The live migration process requires transfer of the complete states of a VM from source host to destination host. The complete state of VM includes all the resources that are used in the source host by the VM such as volatile storage, permanent storage, internal state of the virtual CPUs, and

network state. The network attached storage provides the permanent storage in the data centre, so, it is not required to move the permanent storage during the migration of VM. The internal states of the virtual CPUs are very small amount of data which can be only a few kilobytes in size, so it does not take a considerable amount of time to be transferred. Longer periods are required to transfer the volatile memory contents which affect the performance of the live migration process. More attention is given therefor to improve the transfer of volatile memory from the source to the destination [3].

The live migration process involves four main stages:

1. Setup stage: This stage involves selecting the migrated VM along with destination physical host. It also involves setting up a transfer control protocol (TCP) connection to migrate VM's configuration data between the source and the destination host. Finally, during this stage the memory is allocated and the skeleton of the VM is set up on the destination physical host.
2. Memory transfer stage: This stage includes the pre-copy of memory of the migrated VM to the new allocated memory on the destination host.
3. Storage transfer stage: This stage includes transfer of an up-to-date copy of the virtual hard disks from the source physical host to the destination physical host.
4. Network clean-up stage: This is the last stage of live migration process which involves updating all network switches to make sure that all the connections that were opened before the migration remain opened after the migration.

The performance of live migration of VM is measured by four metrics. They are as follows:

1. Downtime: This is the time period during which the VM is completely shut down.
2. Total migration time: This is the time period between the start of live migration until the resource of the source host are released.
3. Time-to-responsiveness: This represents the time span after the resume phase has ended until the VM achieves a certain guaranteed minimal utilization.

4. Amount of transferred data: This is the measurement of the amount of data received at the destination host from the different sources.

The benefits of live migration are as follows:

1. Live VM migration finds its importance in load balancing among different physical servers. When load of physical server increases, then VMs running onto it will be migrated to lightly loaded servers for load balancing.
2. Transparent movement of Virtual Machine.
3. Live migration removes the problem of residual dependencies of Virtual machines [4].

## **1.4 Automating Live Migration Process**

When the number of customers increases, the capacity of data centre should be increased by increasing the physical infrastructure to accommodate the increased load. The addition of more physical servers and other infrastructure is a continuous process in the data centre. So, managing the entire datacentre running with large number of physical servers and virtual servers becomes cumbersome task. The manual migration of VM to different host becomes difficult task in case there is sudden change in workload. So, the live migration process should be automated which helps to balance the workload in the data centre by distributing load to different system uniformly and avoids performance degradation maintaining SLA with the customers.

First and foremost, priority of any organisation is to make sure that their customers are happy with their strategy and trusting them, because customer is their source of income. So, organisations always provide best and efficient techniques for their customers. The idea is to identify overloaded host and underutilised host then pick a virtual machine from overloaded host and migrate it to underutilised one having enough physical resources to accommodate it [4].

## **1.5 Load Balancing**

Load balancing in the cloud is the process of distributing workloads and computing resources efficiently across multiple computers, networks or server clusters to ensure that the resources are optimally assigned and utilized. To achieve the load balancing condition on the cloud, there are

load balancing algorithm and live migration of VMs algorithm which are discussed in more detail in the methodology section. The load balancing algorithm incorporates CPU and RAM scaling of the cloud infrastructure and if it becomes inevitable then migration of VMs occurs to achieve load balancing condition in the cloud.

## 1.6 Problem Definition

In Cloud Computing, dynamic resource allocation and load balancing is an interesting issue which is open for research. The success of this rising model is dependent on the effectiveness of techniques used to allocate the resources in most optimal way. To optimally allocate resources to virtual machines and perform load balancing, need to determine a precise estimation of the load that a particular virtual machine can handle. As the virtual machines seek more resources on the go, the load experienced by the virtual machines increases exponentially. In some situation, the virtual machine may be under-utilizing the allocated resources, in that situation, scaling down of the resources is necessary. If the virtual machine is overloaded, in that situation scaling up of resources is needed. If the virtual machine is overloaded and is beyond the scope of scaling, migration of that particular virtual machine to a new host node will have to be performed which can accommodate the resource demand of that virtual machine.

For the above problem definition, the following scenario has been taken up as the problem statement.

Let  $C = \{S_1, S_2, \dots, S_n\}$ , where  $C$  is a cloud and  $S_1, S_2, \dots, S_n$  are the servers. Let  $S_j = \{V_{j1}, V_{j2}, \dots, V_{ji}\}$  where  $V_{j1}, V_{j2}, \dots, V_{ji}$  are the virtual machines in the server  $S_j$ . Let  $V_{ji} = \{V_{id}, V_{cpu}, V_{ram}\}$  where  $V_{id}$  is the virtual machine Id,  $V_{cpu}$  is the speed of the processor and  $V_{ram}$  is the RAM size of the virtual machine. Let  $CPU_{maxi}$  and  $RAM_{maxi}$  be the changeable maximum CPU and RAM respectively that can be allocated to the virtual machine  $V_i$ . Let  $R_1, R_2, R_3, \dots, R_m$  be the response time of the applications running in virtual machines and  $m$  is the total number of virtual machines in the cloud. The problem is to allocate the right amount of  $CPU_{sci}$  and  $RAM_{rmi}$  with the help of scaling and migration to each virtual machine  $V_i$  to ensure that the response time  $R_i$  is within the acceptable range [5].

## **1.7 Objectives**

The objectives of this research to improve the performance of applications running inside virtual machine using dynamic resource scaling and automated live migration of virtual machine in the cloud. The main objectives are listed below:

1. Implement load balancing in the Cloud through dynamic resource scaling and live migration of Virtual Machines.
2. Performance analysis of Virtual Machines after load balance implementation in the Cloud.

## CHAPTER 2

### LITERATURE REVIEW

There has been some research work towards resource allocation and load balancing in the cloud. In paper [5], authors present an algorithm which dynamically and efficiently allocate resources based on the need and distribute the load across the servers. They presented an architecture which consists of Xen Cloud Platform and they have used response time of virtual machines as a metric for the algorithms. The authors showed that the proposed algorithms improved the performance of applications running in virtual machines significantly by using the feature scaling and live migration. In paper [6] author presents an architecture and algorithm named compare and balance. The architecture uses a cron job to execute scripts on each host to monitor the resources like CPU and IO load, and log their usage to the log directory on the shared storage. The algorithm runs on each physical machine and dynamically migrates virtual machines from one node to another node based on resource usage. In paper [7] author presents a mechanism for load balancing of virtual machine resources based on genetic algorithm. The author considers historical data and current state of the system and uses tree structure to mark the chromosome of genes and every mapping solution is considered as one tree. Here scheduling node of the system in the first level is the root node, all nodes in the second level represents physical nodes and all nodes in the third level represents virtual machines. In paper [8], the author presents a mechanism based on weighted least connection algorithm. For this, author considers the webservers running long-connectivity applications. Exponential smoothing forecasting method is used as prediction algorithm and it takes historical data and distinguishes them through the smoothing factor to let recent data make a greater impact on the predictive value than long-term data. In paper [9] authors present dynamic and integrated resource scheduling algorithm for Cloud datacentres. Here author considers the factors like CPU, memory and network bandwidth and develops an integrated measurement for the total imbalance level of a cloud datacentre as well as the average imbalance level of each server. In paper [10] author presents a load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. In paper [11], author presents a dynamic load balancing algorithm based on virtual machine migration. The algorithm uses trigger strategy based on fractal methods. The strategy determines the timing of the virtual machine migration through forecasting.

## **CHAPTER 3**

### **METHODOLOGY**

This chapter describes the methodology used in order to achieve the objectives of this dissertation work. For the problem that has been mentioned in the problem statement section, algorithms tested in XenServer cloud environment which was setup in both dedicated hardware and in the virtualized environment.

#### **3.1 Xen**

Xen is a hypervisor which allows the execution of multiple virtual machines on a single physical machine. It is responsible for CPU scheduling of the all virtual machines running on the hardware [5]. Xen abstracts the hardware for the virtual machines and also controls the execution of virtual machines. Domain U (DOM U) are the virtual machines which has no direct access to physical hardware on the machine. Domain 0 (DOM 0) is a virtual machine running on the Xen hypervisor which has special rights to access physical I/O resources as well as interact with the other virtual machines (DOM U). All Xen virtualization environments require DOM 0 to be running before starting any other virtual machines.

#### **3.2 CPU Scheduler in Xen**

Xen has a variety of CPU schedulers in the past and Xen developers went through various CPU scheduling algorithms. But among various algorithms, three algorithms: BVT (Borrowed Virtual Time), SEDF (Simple Earliest Deadline First), and Credit scheduler utilized for a longer time due to their efficiency.

##### **3.2.1 Borrowed Virtual Time (BVT)**

This algorithm was introduced in Xen 2.0 which is a virtual time-based fair-share CPU Scheduler for Xen 2.0 and Xen 3.0 and the shares of CPU time were determined by their weights which is known as context switch allowance.

##### **3.2.2 Simple Earliest Deadline First (SEDF)**

This algorithm was introduced in Xen 3.0 which provides weighted CPU sharing in an intuitive way and uses real-time algorithms to ensure time guarantees but it was removed from Xen 4.6.

##### **3.2.3 Credit Scheduler**

The Credit scheduler is a fair share scheduler designed to divide CPU time between domains by



relative weight. It works in conjunction with SMP hardware to distribute load amongst multiple virtual CPUs abstractions or VCPUs. While using the credit scheduler, each physical CPU manages a queue of VCPUs which sorted by priority. Priority calculated with the number of credits consumed by a given VCPU. The administrator defines credits. A VCPU can execute while it is under its limit. When all credits have been consumed, it is considered over its limit. When credits are balanced, it is under the limit [13]. VCPU is nothing but the share of physical CPU (PCPU) provided to guest operating system or domain (VM). The credit scheduler algorithm sets two parameters weight and cap for each guest operating system which are described as below:

**Weight:** Each virtual machine uses CPU with the ratio of weight values. For example: A guest operating system with weight equal to 512 means, it is time to take up physical CPU is twice than guest operating system with weight equal to 256. We can set weight values from a range between 1 and 65535.

**Capacity [Cap]:** Cap is a limit that how much maximum time a guest operating system can get CPU. A guest operating system with cap equal to 100 means it occupies a physical CPU time, a guest operating system with a cap equal to 50 takes only half that time. The default value of cap is 0 which means there is no limit.

### 3.3 System Architecture

The basic architecture consists of N number of XenServers, NFS, requester (client) and cloud controller which is shown in the figure 3.1. Each XenServer has a Xen hypervisor to run multiple virtual machines. The requester sends request for computing resources through cloud controller and the required computing resources are provided in the form of virtual machines.

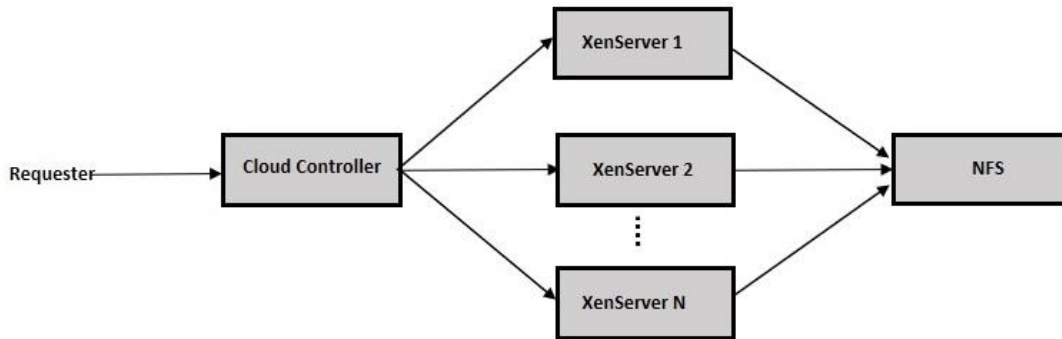


Figure 3.1: System Architecture [5]

### 3.3.1 Resource Scaling Down Activity

The figure 3.2 shows the interaction among various components of Xenserver when applications are running inside a virtual machine. During the interaction, the `DOM_0` sends a `Req_ResourceStatus` message to the virtual machine, requesting the details of resource usage. If the resources granted to the virtual machine is underutilized, the virtual machine sends a `Res_Underloaded` message back to the `DOM_0` via the hypervisor. The `DOM_0` makes use of the existing scaling methods to indicate the level to which the resources must be scaled down. The information provided by the `DOM_0` is then sent to the hypervisor which performs the job of scaling down of the resources, the result of which is reflected in the operation of the virtual machine.

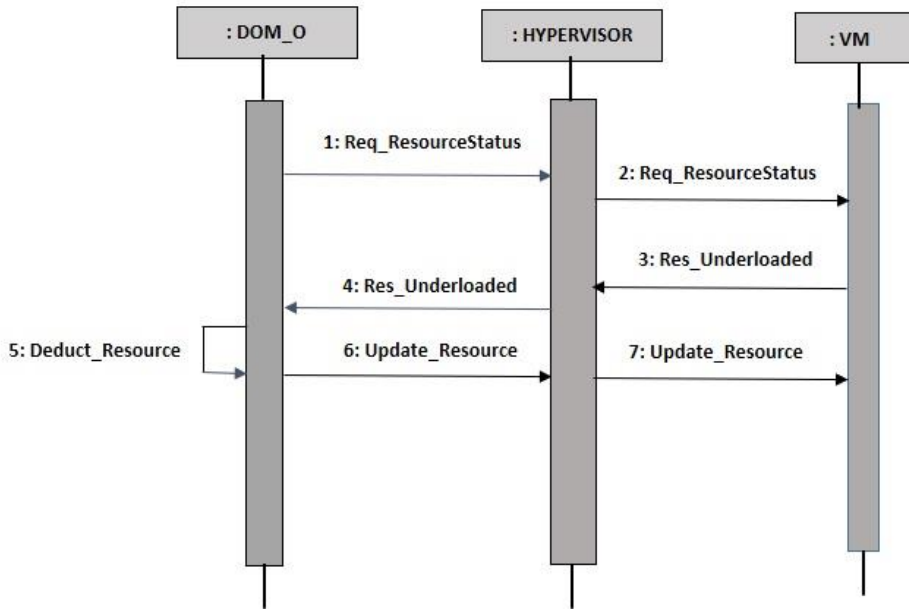


Figure 3.2: Sequence of activities to scale down the resources [5]

### 3.3.2 Resource Scaling Up Activity

The figure 3.3 shows the interaction among various components of Xenserver where the virtual machine is overloaded, and which requires more resources to accommodate the load. The `DOM_0` sends a `Req_ResourceStatus` message to the virtual machine through the hypervisor requesting the details of resource utilization. The virtual machine sends a `Res_Overloaded` message back to the `DOM_0` stating that resources are overloaded. The `DOM_0` searches for resources that are not used or wasted by other virtual machine in the cloud pool. When resources are found, the resources are allocated to the virtual machine as per requirements which is the process of scaling up of resources.

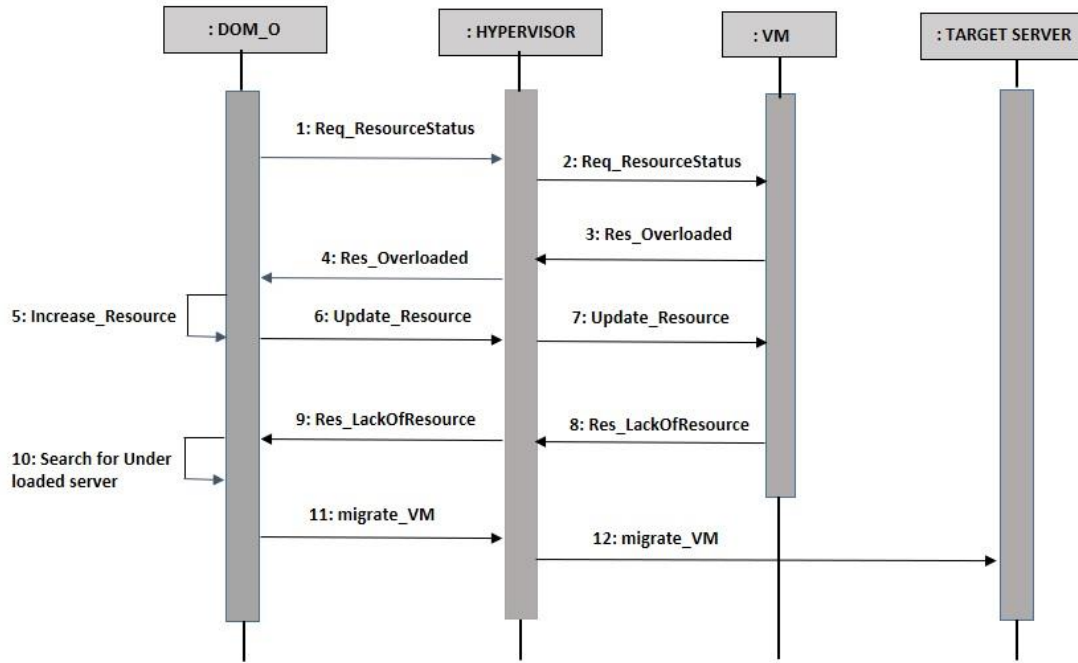


Figure 3.3: Sequence of activities to migrate VM to suitable target server [5]

### 3.3.3 Live Migration Activity

In case if there are no enough resources that can be provided in the current Xenserver, the virtual machine needs to be migrated to a suitable server. The DOM\_0 identify the suitable target server in the cloud pool and migrates the virtual machine to that target server. The server to which the virtual machine is migrated is decided based on the selection criteria provided in the algorithm. Based on this, hypervisor initiates the migration of the virtual machine to the suitable target server.

## 3.4 Algorithm

There are two algorithms which perform resource allocation, resource scaling and live migration of virtual machine which are described as below.

### 3.4.1 Load Balancing and Live Migration Algorithm

The Load Balancing algorithm verifies if the CPU usage and RAM usage are in the defined limits. After the verification process is completed, the algorithm initially scales the resources in an upward or downward manner based on resource availability. If the resources required for scaling are not available, then algorithm performs live migration of relevant virtual machines from that specific

XenServer node to another suitable XenServer node.

This algorithm [5] combines the processes of scaling and migration by calling the relevant functions.

Input: CPU utilization

Output: Scaling or Migration

**Algorithm:**

Amt  $\leftarrow$  Resource in Percentage

T  $\leftarrow$  Time in second

CPU  $\leftarrow$  CPU Utilization

RAM  $\leftarrow$  RAM Utilization

**while** VM is running **do**

**if** CPU > UpperThreshold1 and CPU < UpperThreshold2

**then**

            Scale Resource up by Amt

            Wait for duration T

**if** CPU > UpperThreshold1 and CPU < UpperThreshold2

**then**

            Scale Resource up by Amt

            Wait for duration T

**if** CPU > UpperThreshold1 and < UpperThreshold2

**then**

            MigrateToServer( )

**end if**

```

end if

else if CPU > UpperThreshold2 then

    Scale Resource up by 2*Amt

if CPU > UpperThreshold2 then

    MigrateToServer( )

end if

else if CPU < LowerThreshold then

    Scale Resource down by Amt

    Wait for duration T

if CPU < LowerThreshold then

    Scale Resource down by Amt

    Wait for duration T

    if CPU < LowerThreshold then

        MigrateToServer

    end if

end if

end if

if RAM > UpperThreshold1 and RAM < UpperThreshold2

then

    Scale Resources up by Amt

    Wait for duration T

```

**if** RAM > UpperThreshold1 and RAM < UpperThreshold2

**then**

Scale Resource up by Amt

Wait for duration T

**if** RAM > UpperThreshold1 and < UpperThreshold2

**then**

MigrateToServer( )

**end if**

**end if**

**else if** RAM > UpperThreshold2 **then**

Scale Resource up by 2\*Amt

**if** RAM > UpperThreshold2 **then**

MigrateToServer( )

**end if**

**end if**

**end while**

### **3.4.2 Algorithm for Scaling**

This algorithm [5] performs the process of scaling of CPU or RAM according to the input parameters supplied to it. The credit scheduler is used to scale the CPU and Xen API based command `xe vm-param-set` has been used to scale RAM as required.

**Algorithm:**

Amt  $\leftarrow$  Resource in Percentage

**if** Resource Name = CPU and Scale Type = UP **then**

```

Cap ← Current CPU cap value of VM

if Cap < 100 then

    Scale up the CPU by Amt

end if

else if Resource Name = CPU and Scale Type = DOWN

then

    Cap ← Current CPU cap value of VM

    if Cap < 20 then

        Scale down the CPU by Amt

    end if

else if Resource Name = RAM then

    SMin ← Static min value of RAM

    SMax ← Static max value of RAM

    DMin ← Dynamic min value of RAM

    DMax ← Dynamic max value of RAM

    if Scale Type = DOWN and DMax < SMin then

        Scale down RAM by Amt

    else if Scale Type = UP and DMax < SMin then

        Scale up RAM by Amt

    end if

end if

```



# CHAPTER 4

## IMPLEMENTATION

This chapter describes about the implementation details of the methodology presented in the previous chapter. Experiment was performed setting up Xen Cloud platform in physical hardware and in the virtualized environment in virtual box as Xenserver supports Para-virtualization.

### 4.1 Experimental Set-up

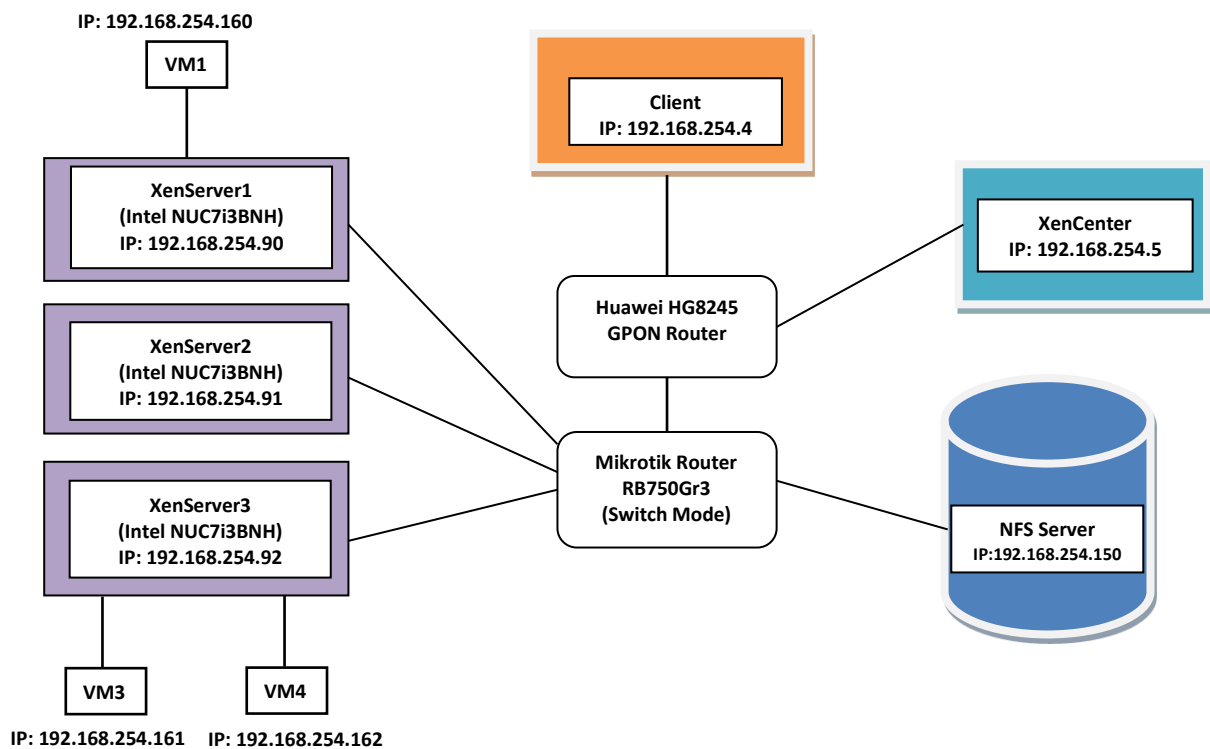


Figure 4.1: Experimental set up of XenServer cloud platform in physical hardware

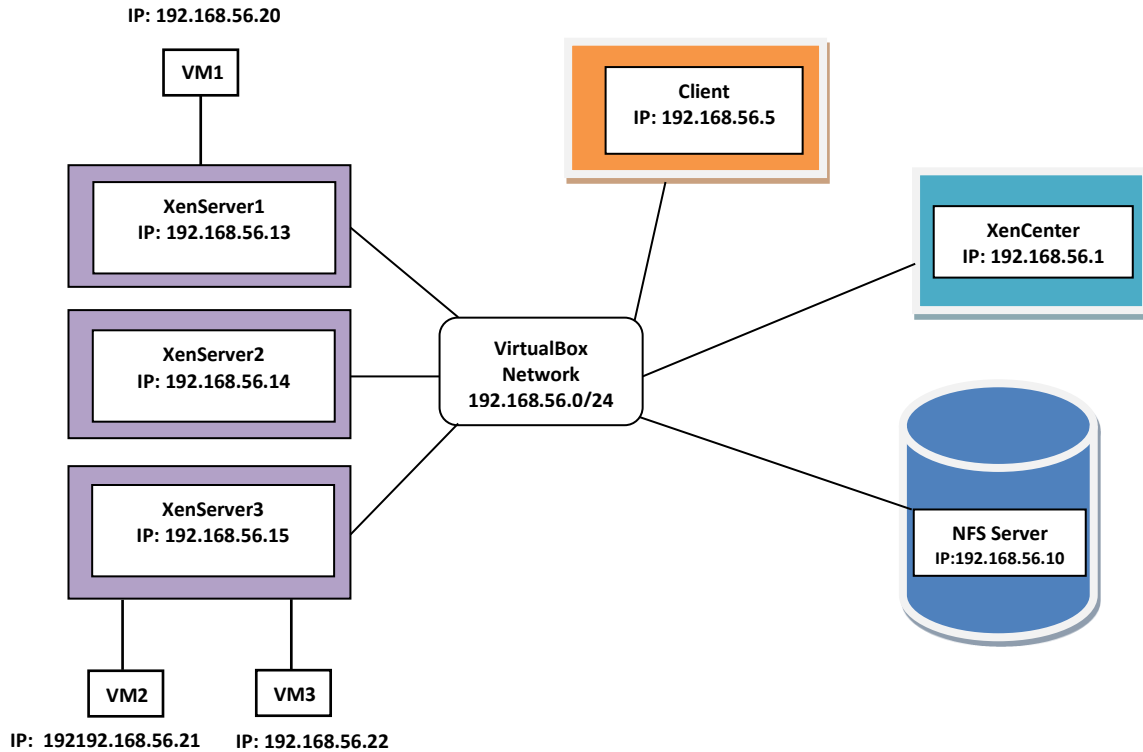


Figure 4.2: Experimental set up of XenServer cloud platform in virtual environment

Figure 4.1 shows XenServer cloud architecture setup in physical hardware. The set up consists of three XenServers installed on three identical Intel NUC hardware having configuration Intel i3 7<sup>th</sup> generation 2.40 Ghz processor, 8GB RAM on two servers and 4GB RAM in one server. NFS is created on another machine (Dell Laptop) having configuration Intel i3 6<sup>th</sup> generation 2.0Ghz processor, 1TB storage and 4GB RAM with CentOS 7 operating system. The setup also consists of XenCenter enabled system to manage XenServer and Virtual Machines through GUI which was installed in another machine (Dell Laptop) having configuration Intel i5 6<sup>th</sup> generation processor with Windows 10 operating system. Another machine MacBook Pro with configuration Intel i5 processor having MacOS as operating system was used as a client test machine. All the above machines were connected to a network 192.168.254.0/24 by using Huawei HG8245 GPON Router and Mikrotik Router RB750Gr3 in switch mode as shown in the figure 4.1. Three virtual machines having configuration of 1GB RAM and 10GB hard disk were created and assigned to Xenservers as shown in the figure 4.1 above.

The figure 4.2 shows XenServer cloud architecture setup in virtualized environment. The setup is exactly same as the physical one, but the environment was setup in the single machine having configuration intel i3 7<sup>th</sup> generation processor, 12GB of RAM and Windows10 as operating system. In the system, VirtualBox was installed and on top of that three XenServers, one NFS server, three VMs, one client test VM were setup. Also, XenCenter was installed in the same Windows10 host and all the servers were interconnected in the network of 192.168.56.0/24 provided by VirtualBox. The details of resource allocation are given in the table 4.2 and table 4.3 for physical and virtualized environment setup.

In both test environment, the load balancing algorithm is stored locally in the Xenservers and begin execution as when load is generated on the test VMs. The load balancing algorithm, then decides on whether to scale the resources for the virtual machines or decides on live migration of virtual machines if necessary, thereby reducing the response time of the virtual machines. The availability of resources is checked by a Xen api called xentop.

## **4.2 System Specifications**

The system specifications used to establish the experimental setup are as follows:

Hardware Model:

1. Intel NUC: Intel NUC7i3BNH, Processor: Intel® Core™ i3-7100U CPU @ 2.40GHz, RAM: 8GB DDR4, Hard Disk: 1TB
2. Dell Inspiron 3467: Processor: Intel® Core™ i3-6006U Processor @2.00 GHz, RAM: 4 GB RAM, Hard Disk: 1TB
3. Dell Inspiron 5559: Processor: Intel Core i5-6200U Processor @2.3Ghz. RAM: 8GB, Hard Disk: 1TB
4. Apple MacBook Pro: Processor: Intel Core i5 processor @ 2.3Ghz, RAM: 8GB, Hard Disk: 256GB SSD
5. Huawei HG8245 GPON Router
6. Mikrotik Router RB750Gr3

From the above hardware system, following resources are assigned to XenServer Platform as shown in the table below:

Table 4.1: Resource Allocation for XenServer Platform Setup on dedicated hardware

S.N.	Host	IP Address	Disk (GB)	RAM (GB)	CPU	OS
1	XenServer1	192.168.254.90	1000	8	Intel® i3-7100U	XenServer7.0
2	XenServer2	192.168.254.91	1000	8	Intel® i3-7100U	XenServer7.0
3	XenServer3	192.168.254.92	500	4	Intel® i3-7100U	XenServer7.0
4	NFS	192.168.254.150	1000	4	Intel® i3-6006U	CentOS 7.3
5	Client	192.168.254.4	256	8	Intel Core i5	MacOS
6	XenCenter	192.168.254.5	1000	8	Intel® i5-6200U	Windows 10

Table 4.2: Resource Allocation for XenServer Platform in VirtualBox

S.N.	Host	IP Address	Disk (GB)	RAM (MB)	CPU (Virtual)	OS
1	XenServer1	192.168.56.13	65	2048	Intel® i3-7100U	XenServer7.0
2	XenServer2	192.168.56.14	65	2048	Intel® i3-7100U	XenServer7.0
3	XenServer3	192.168.56.15	65	2048	Intel® i3-7100U	XenServer7.0
4	NFS	192.168.56.102	65	512	Intel® i3-7100U	CentOS 7.3
5	VM1	192.168.56.20	10	512	Intel® i3-7100U	CentOS 6.9
6	VM2	192.168.56.21	10	512	Intel® i3-7100U	CentOS 6.9
7	VM3	192.168.56.22	10	512	Intel® i3-7100U	CentOS 6.9
8	Client VM	192.168.56.5	10	512	Intel® i3-7100U	CentOS 7.3
9	XenCenter	192.168.56.1	500	3584	Intel® i3-7100U	Windows 10

## 4.3 Oracle VM VirtualBox

VirtualBox is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop and embedded use [17]. It is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as for home lab environment use which is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. It runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems. It has a feature of extending the capabilities of computer so that it can run multiple OSes, inside multiple virtual machines, at the same time.

## 4.4 XenServer

### 4.4.1 Introduction to XenServer

The XenServer is an open source virtualization platform that allows the building of both private and public clouds. The XenServer includes Xen Hypervisor, selected Xen API based tools and an integrated solution package for storage and networking in cloud. The additional functionality offered by XenServer platform are:

- i. **Virtual Machine Lifecycle:** This functionality includes the features of screen snapshots, migration of the virtual machines and creation of checkpoints.
- ii. **Resource Pools:** Resource pools can be created, thereby enabling flexible storage and networking.
- iii. **Event Tracking:** Particular events of interest can be tracked by using the progress and the notification features.
- iv. **Performance Monitoring:** The XenServer offers real time performance monitoring and alerting.
- v. **XenMotion Live Migration:** XenMotion is a feature than enables the migration of a virtual machine from one server to another server. This functionality also enables cross pool migration.

## 4.4.2 Installing XenServer in Physical Hardware and Virtual Box

The installation steps of XenServer on physical hardware and VirtualBox are exactly same except booting process of XenServer OS for the first time while installing. In case of physical hardware, a USB bootable device of Xenserver 7.0 was created and booted it into the machine. In case of VirtualBox, ISO image file of XenServer 7.0 was used and booted it via Virtual CD drive. In both cases, the remaining installation steps are same, so VirtualBox based installation steps are given below:

- i. Create virtual machine with appropriate resource allocation in VirtualBox and start it. Then select XenServer installation image ISO file from computer on virtual CD drive of virtual box which will boot the virtual machine with ISO image.
- ii. After the initial boot messages, the installer does some hardware detection and initialization.
- iii. Then Welcome to XenServer Setup screen is displayed. Choose OK to proceed.

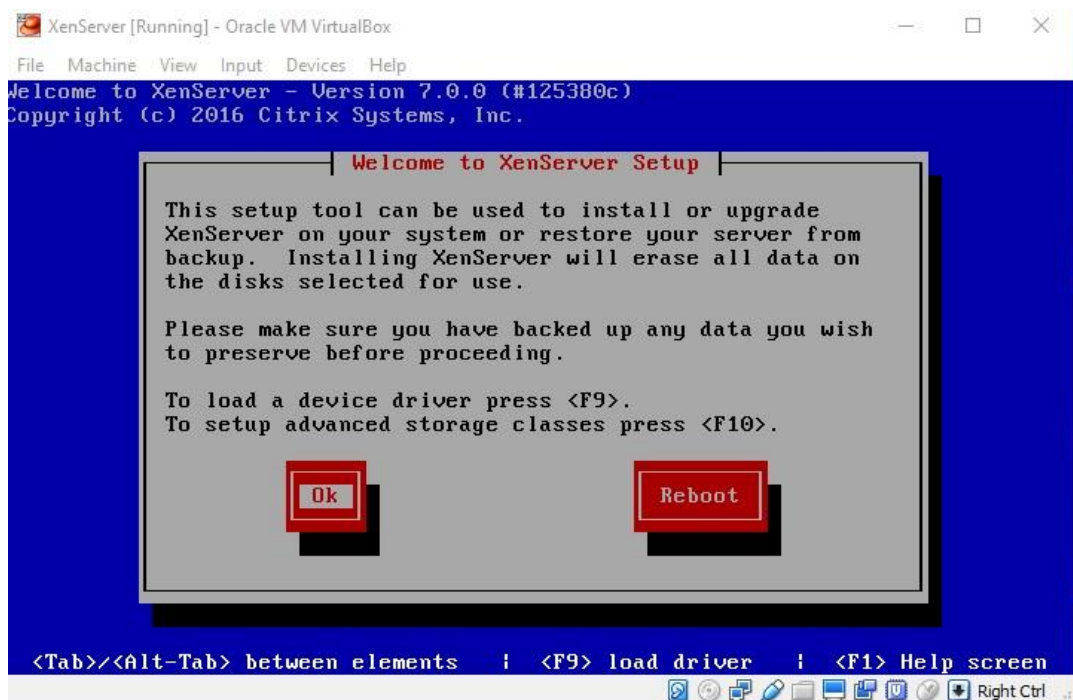


Figure 4.3: XenServer Installation Start Console

- iv. The next screen asks to specify the source of the installation packages. Select Local media and choose OK to proceed.

- v. Provide Hostname, IP Address, Subnet, Gateway and DNS for XenServer in Network Configuration Section. Need to provide static IP for XenServer host node.
- vi. In the next prompt, set a root password. Enter the desired password and enter it again to verify it.
- vii. Now installation is ready to proceed, ignore any warning messages which says erasing data from current disk etc. Choose Install XenServer to proceed. A progress bar is displayed as the installation starts.

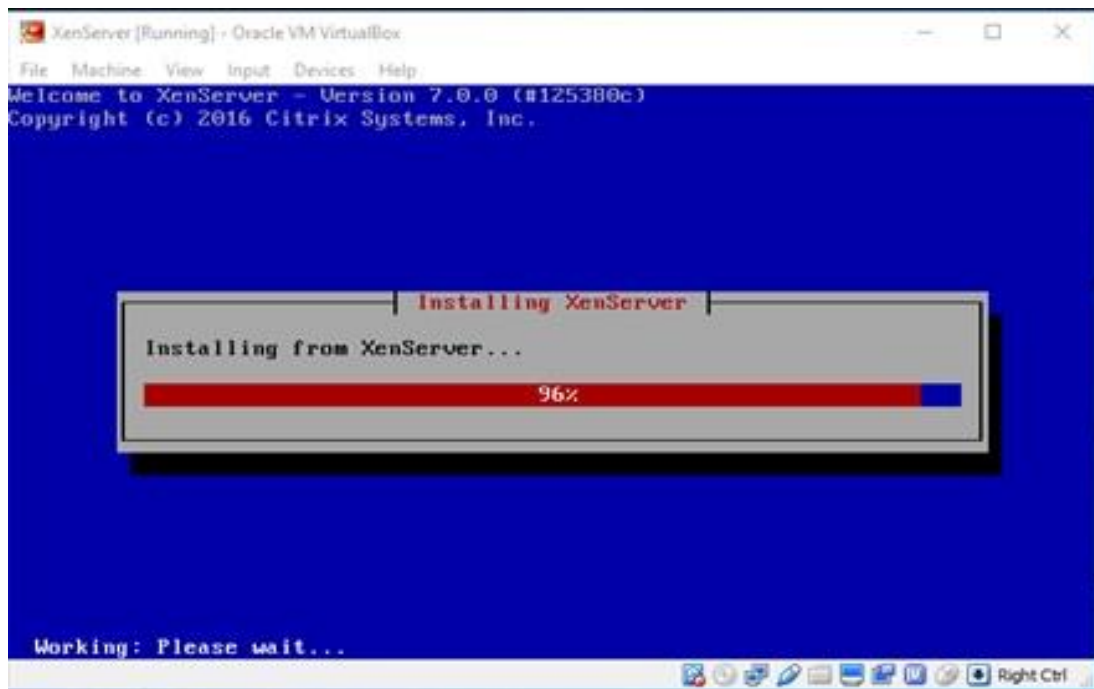


Figure 4.4: XenServer Installation Progress Bar

- viii. Once the installation is completed successfully, then screen is displayed as shown in the figure 4.5. Then need to click OK to reboot server for first boot up process. Hence, installation of Xenserver node is completed. Need to follow same steps to install other XenServer nodes, but IP Address and hostname change for other nodes.

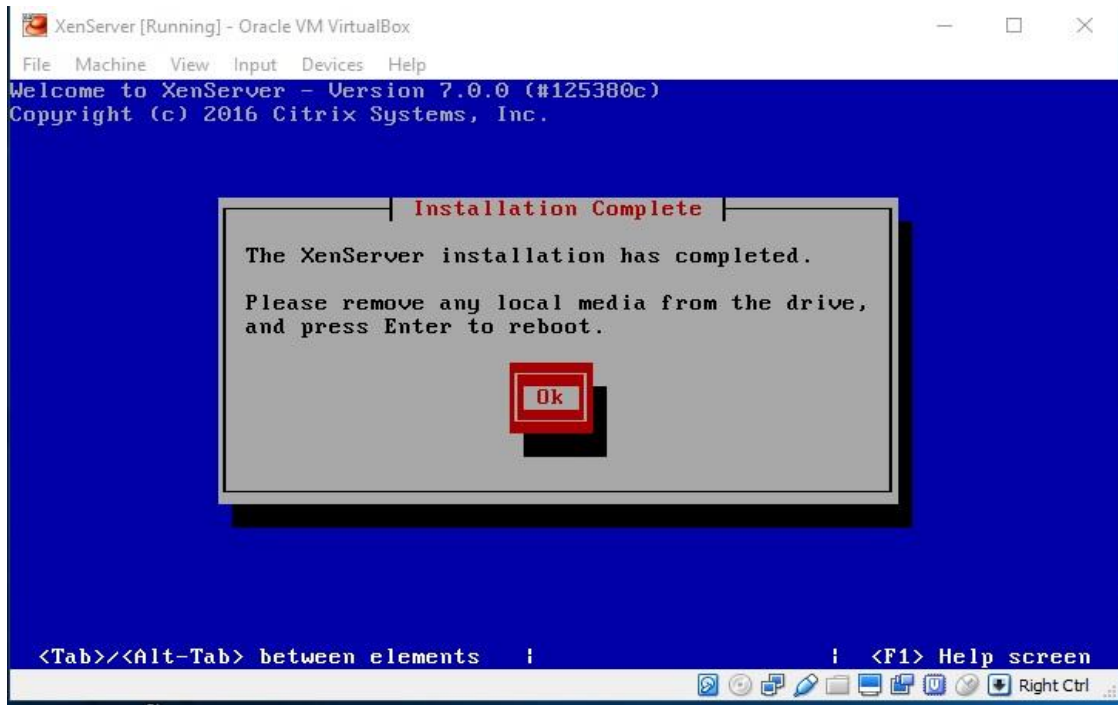


Figure 4.5: XenServer Installation Completion



- ix. The Xenserver Console after booting up looks like as below:

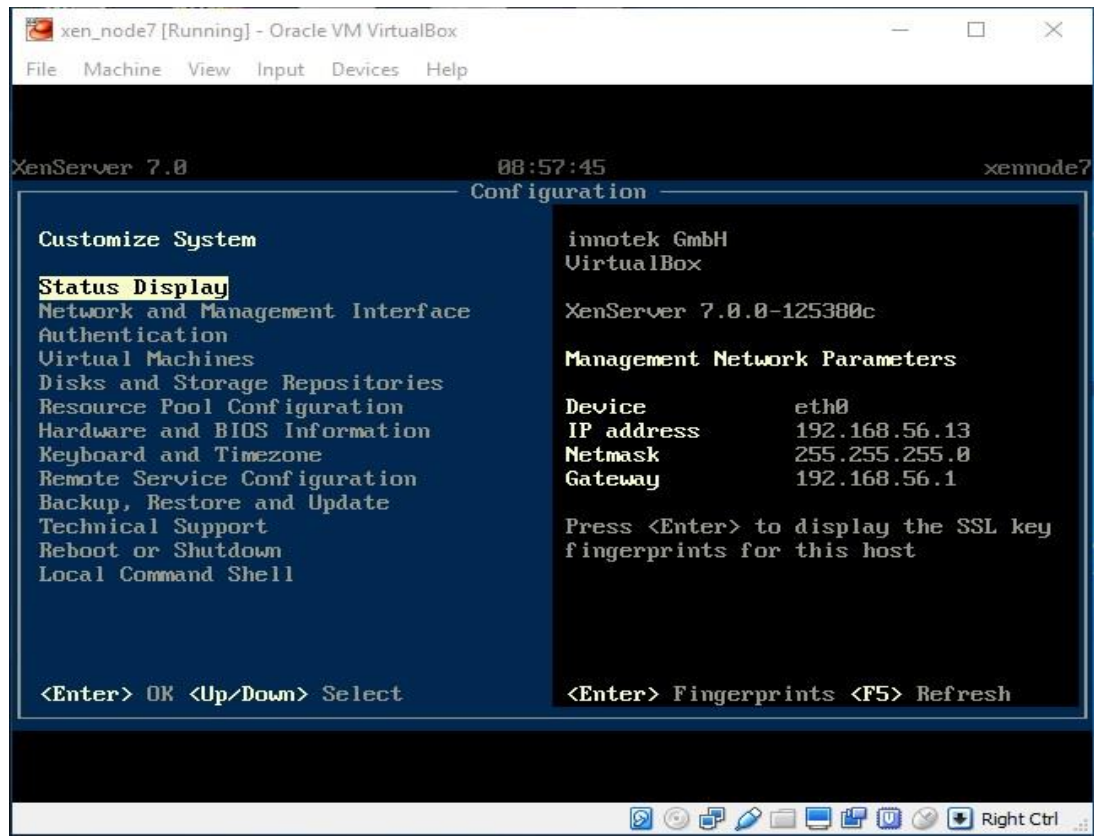


Figure 4.6: XenServer System Configuration Console

- x. The local shell access console looks like shown in figure 4.7.

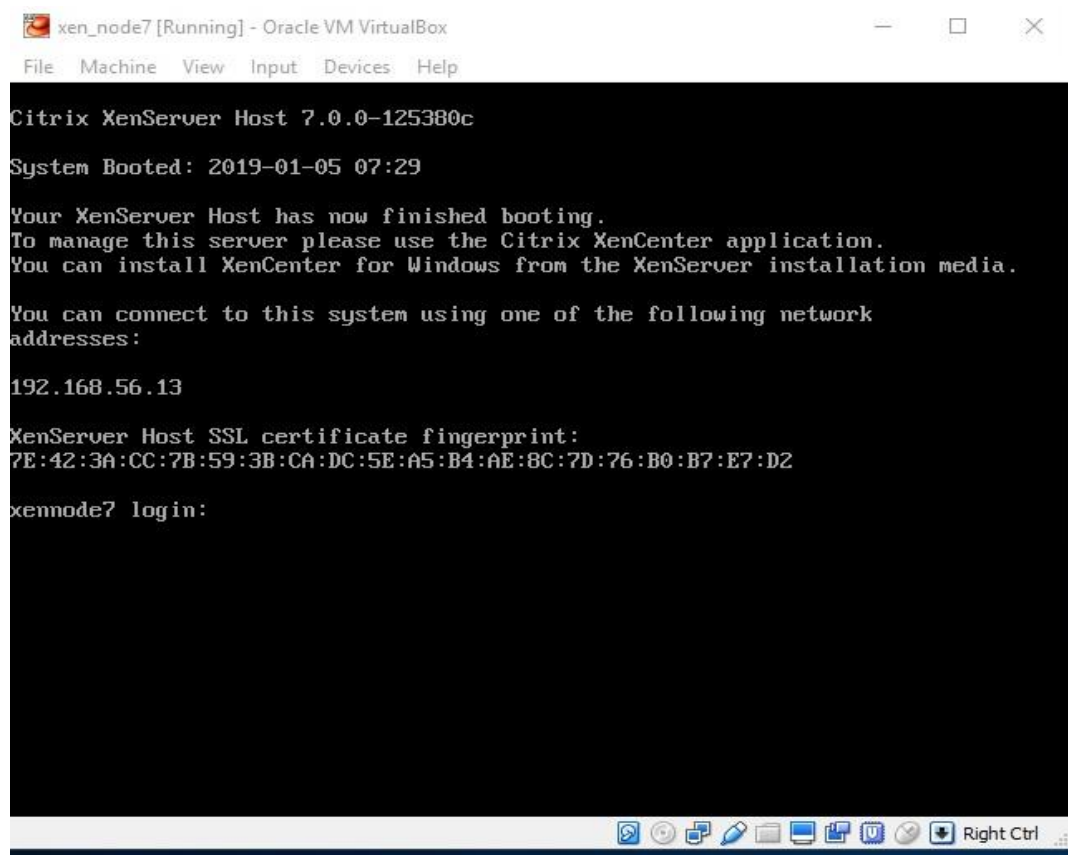


Figure 4.7: Xen Server Login Console

## 4.5 Citrix XenCenter

### 4.5.1 Introduction to XenCenter

Citrix XenCenter is a graphical user interface on the windows platform which is used to manage virtual machines on XenServer. Some of the features of XenCenter include the following:

- Full virtual machine installation, configuration and lifecycle management.
- Dynamic memory management.
- VM snapshot management.
- Full memory snapshots and VM rollback.
- Performance metrics display.
- Long term metrics gathering and analysis.

## 4.5.2 Installation

1. Open the web browser and type IP address of any one Xen Server which will show a webpage with link to download XenCenter.
2. Click on XenCenter Installer link and download it

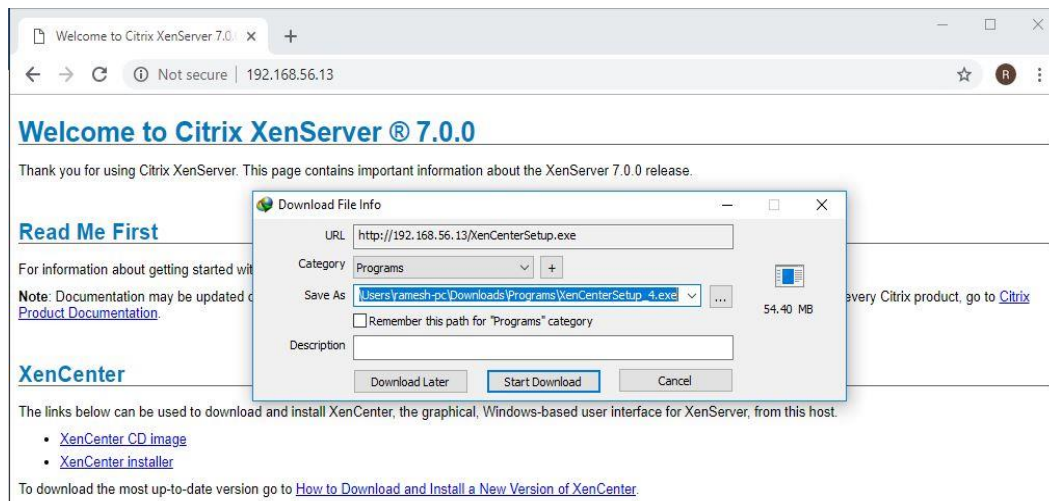


Figure 4.8: XenCenter Download from Xenserver

3. Open downloaded Setup file XenCenterSetup.exe and then follow the steps to install it.
4. There is no specific requirement for installation, just click Next, Next and finish it.

## 4.6 Network File System

### 4.6.1 Introduction to NFS

Network File System (NFS) is a distributed file system protocol which allows the user to access files over a network similar to how files are accessed locally. The NFS has been used as a shared storage for VMs where storage is allocated while creating VM in XenServer. The main reason for implementing NFS shared storage is due to live migration of virtual machines which requires shared storage system.

### 4.6.2 Installation

The operating system used for NFS server was CentOS7.3 which supports NFS. The installation steps are as follows:

As the first step, the following packages were installed on the CentOS server with yum:

```
# yum install nfs-utils
```

Created shared storage for virtual hard disk and ISO images for VMs with following command:

```
#mkdir /nfsfileshare
```

```
#mkdir /iso
```

Shared NFS directories over network so that XenServer and Virtual Machines can access those directories, adding following lines in /etc/exports file:

```
# vi /etc/exports
```

```
/nfsfileshare *(rw, sync, no_root_squash)
```

```
/iso *(rw, sync, no_root_squash)
```

Given appropriate permissions to those shared directories from NFS server so that XenServer can access shared directory:

```
#chmod -R 755 /nfsfileshare
```

```
#chmod -R 755 /iso
```

NFS has been restarted with following command:

```
#systemctl restart nfs-server
```

## **4.7 VM install and Management in XenServer**

Both XenCenter client software and CLI were used for virtual machine installation and management inside XenServer node which are described as below:

### **4.7.1 Using XenCenter GUI (Graphical User Interface)**

- i. Search installed XenCenter App from Windows Search Box and Open it.

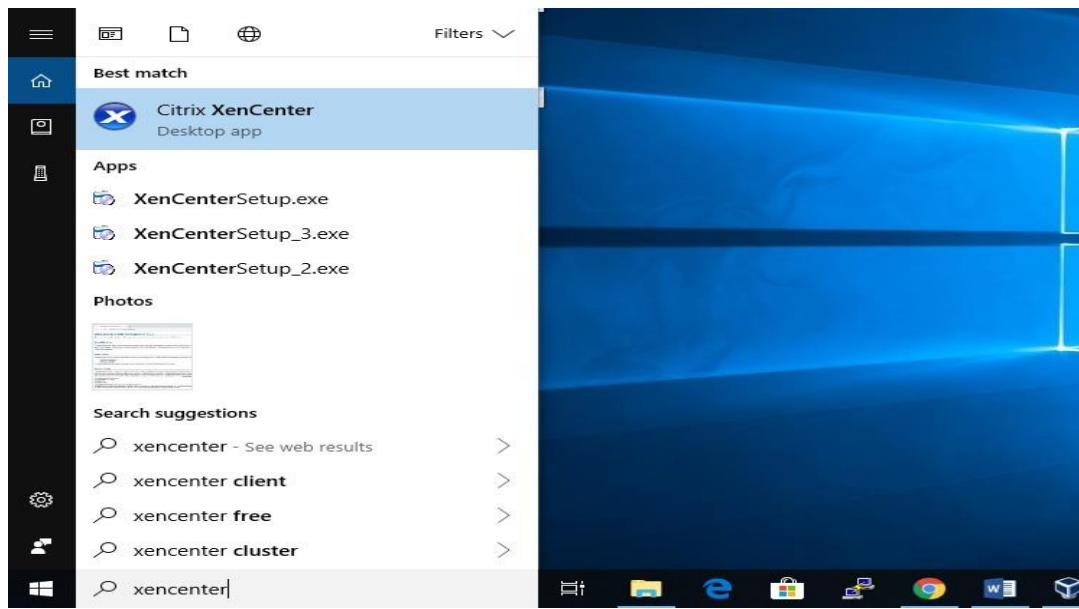


Figure 4.9: Start Citrix XenCenter software

- ii. Go to VM menu from XenCenter and Click on New VM.

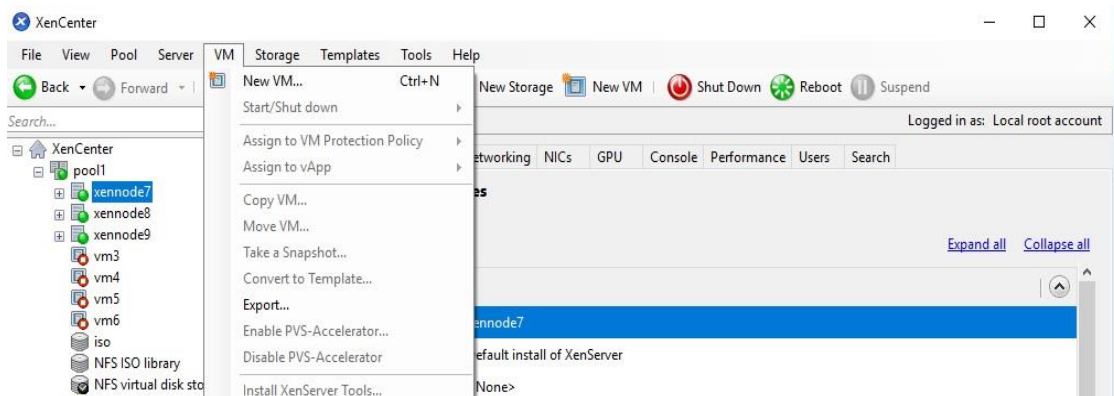


Figure 4.10: XenCenter VM Management Menu

- iii. Select an operating system template for new virtual machine and then click Next.

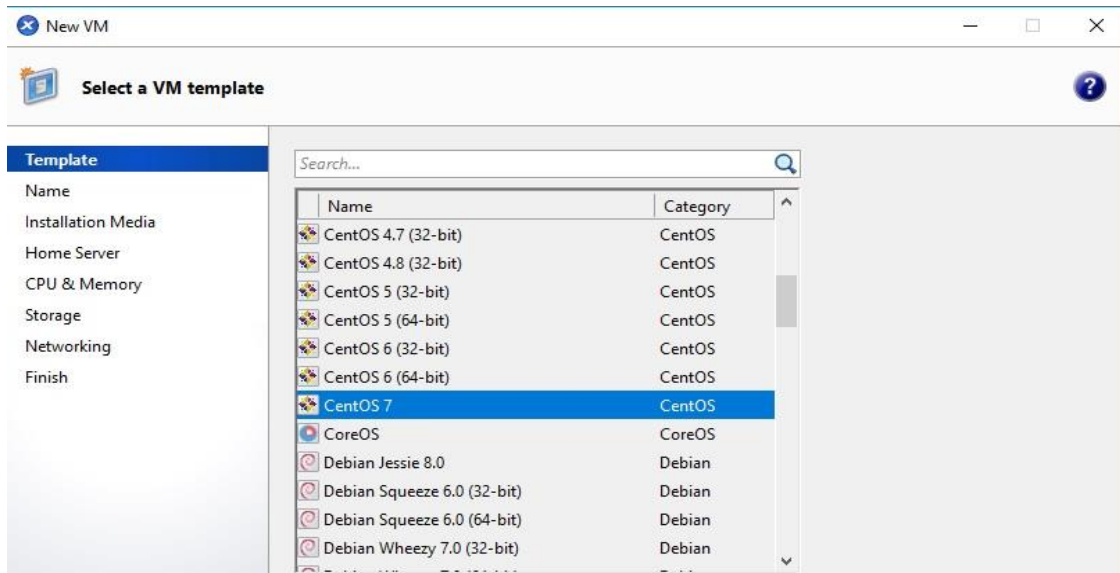


Figure 4.11: VM Selection Template

- iv. Enter a name and description for the new virtual machine.

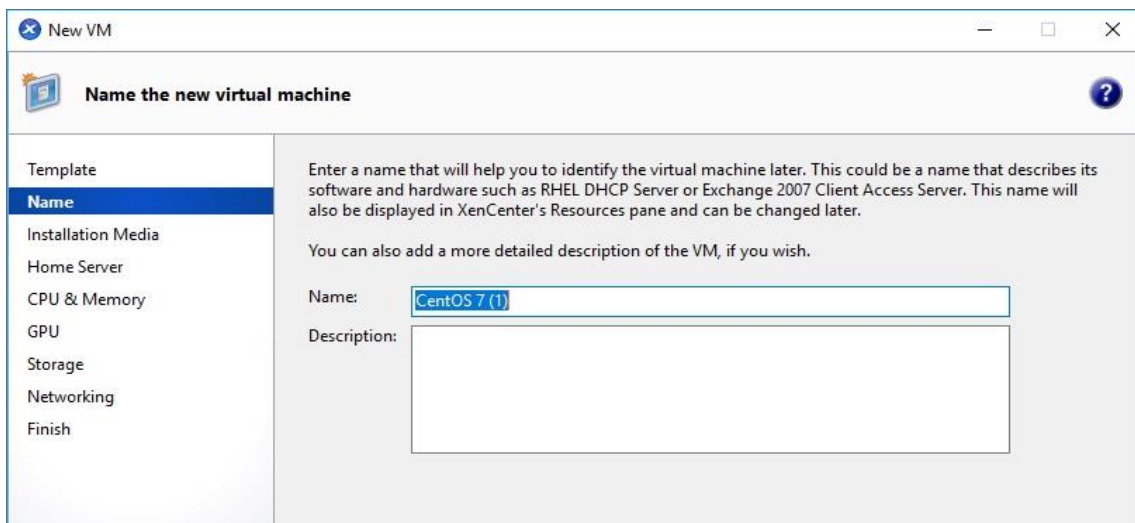


Figure 4.12: Enter the name of the operating system

- v. Enter the location of the guest operating system on installation Media section. The ISO images were setup on NFS, so VM OS is chosen from NFS ISO Library for installation.

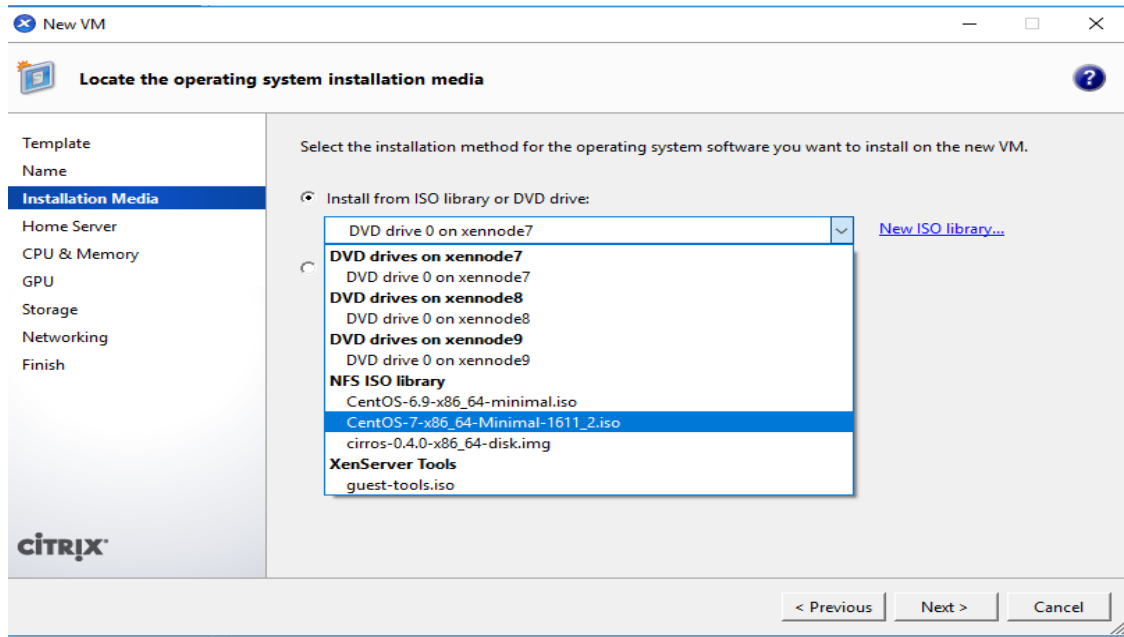


Figure 4.13: Selection of installation media

- vi. Allocate hardware resources: Number of virtual CPUs and size of Memory to a new VM which is being created.
- vii. On the storage section, a hard disk for virtual machine on NFS shared storage is created in NFS server.

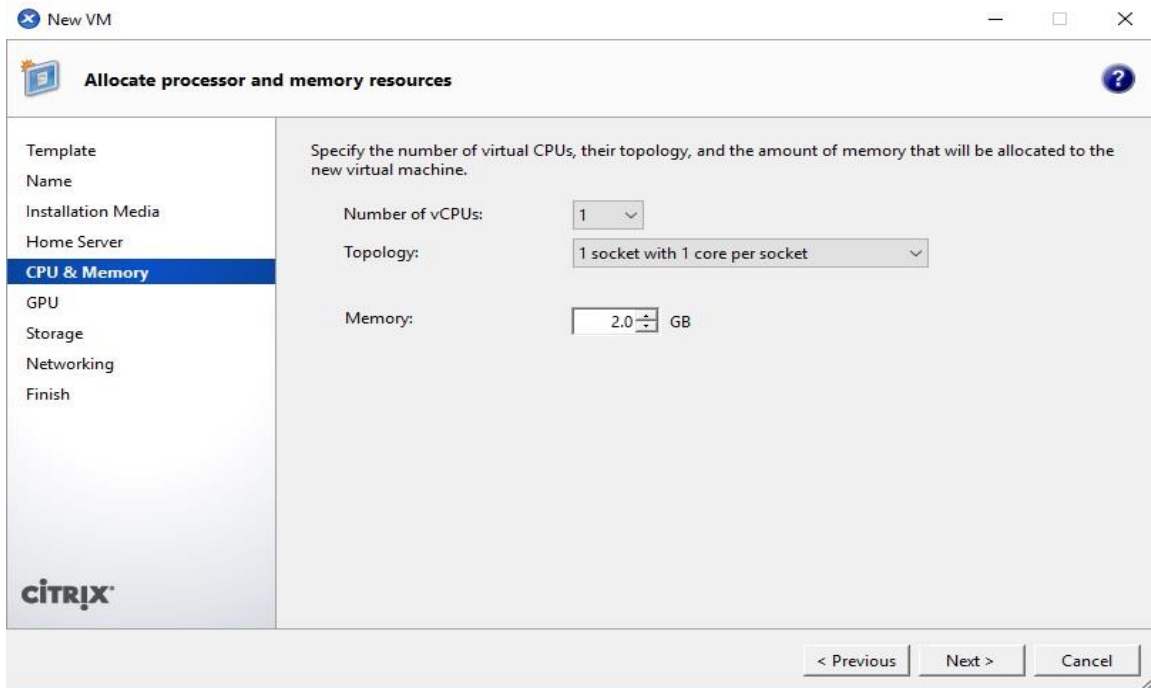


Figure 4.14: CPU and Memory allocation

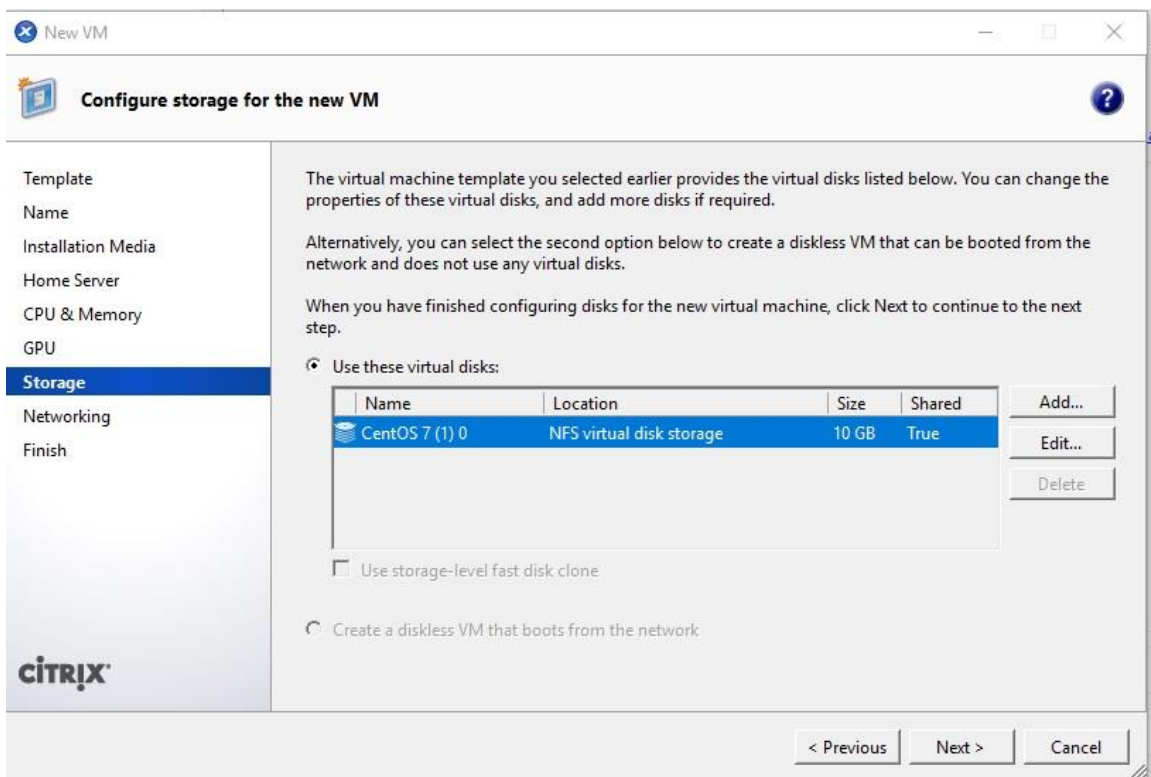


Figure 4.15: Disk information for VM



viii. Add available virtual network interface for new VM. Click Next and then Finish.

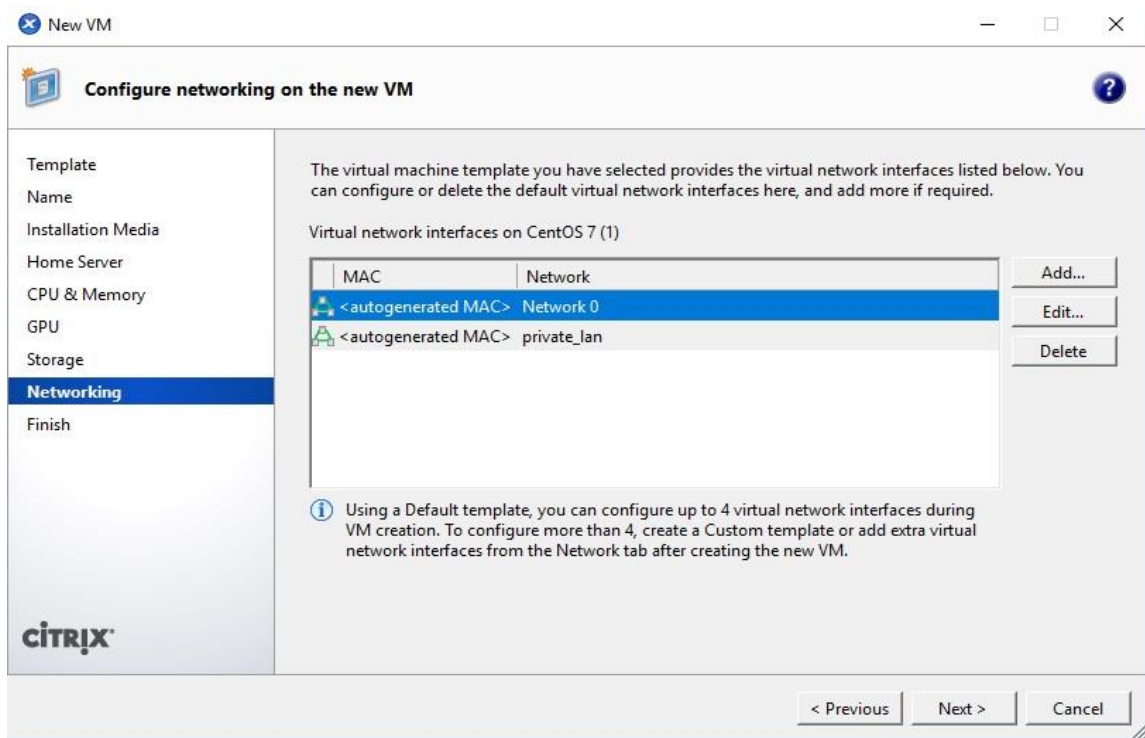


Figure 4.16: Configure Network Settings

ix. New Virtual Machine boots up with provided information for installation. Installation steps depends on Operating System chosen. After installation completion and rebooting VM, XenCenter VM console looks like as below:

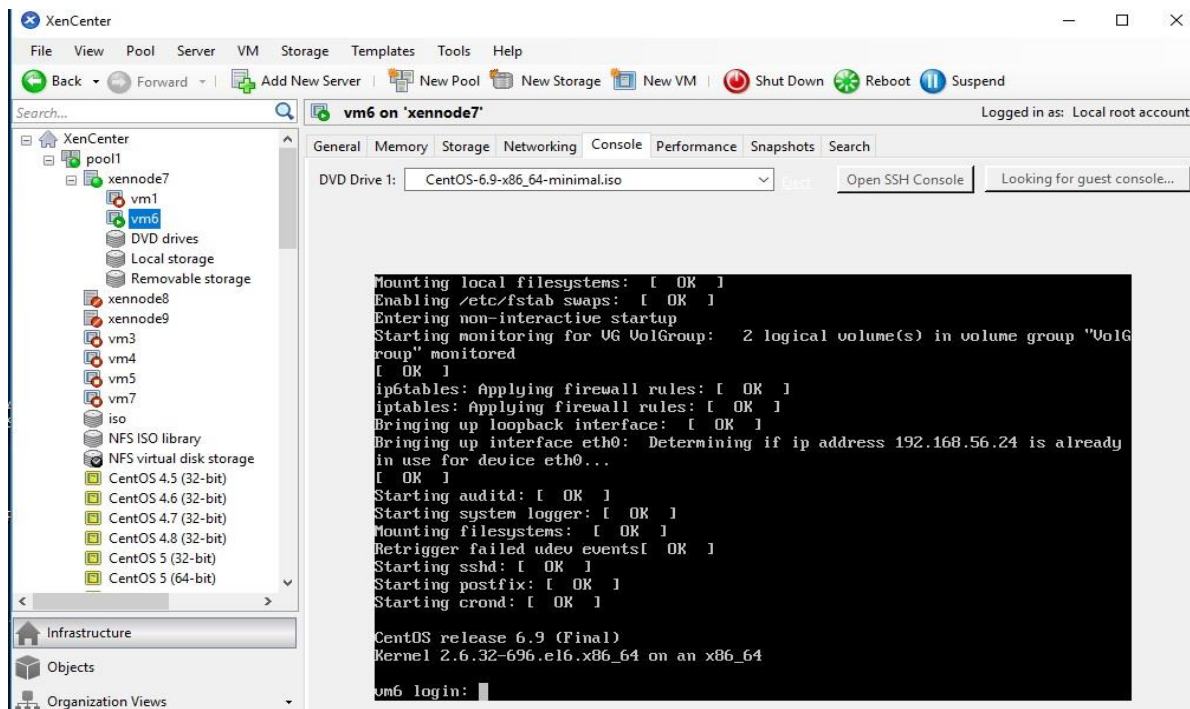


Figure 4.17: VM Console in XenCenter

## 4.7.2 Using CLI (Command Line Interface)

- i. To make an ISO library available to XenServer Hosts, create a NFS shared directory. The NFS server must be set to allow root access to the share.  
`# xe-mount-iso-sr host:/volume`
- ii. Install virtual machine from a template with following command:  
`xe vm-install new-name-label=<VM_NAME> template=<TEMPLATE NAME>`
- iii. List the ISOs available:  
`xe cd-list`
- iv. Place the ISO into the virtual CD drive  
`# xe vm-cd-ad vm=<VM_NAME> cd-name=<NAME_ISO.iso> device=3`
- v. Start and install the OS:  
`# xe vm-start vm=<VM-NAME>`

### 4.7.3 XenServer and VM Management Commands in XenServer

- **xe host-list** - Standard object listing command to list XenServer host objects  
name-label – The name of the XenServer host
- **xe vm-list** - Standard object listing command to list VM objects  
resident-on - The XenServer Host on which a VM is currently resident  
power-state - Current power state; always halted for a Template
- **xl sched-credit** – Used to set or get weight and cap values of VMs  
-d - specify the domain  
-w - specify the weight value  
-c - specify the cap value
- **xe vm-param-get** – Gets the values for all the parameters of the VM  
uuid - unique ID of the VM  
param-name - name of the parameter whose value is required
- **xe vm-param-set** - Sets the values for the specified parameter of the VM  
uuid - unique ID of the VM  
memory-dynamic-max - to set value for dynamic max
- **xe vm-migrate** – Migrate the specified VMs between physical hosts  
vm - uuid of the VM to be migrated  
destination – name of the server from where the VM is migrated  
host - name of the server where the VM is migrated to
- **xentop** - Displays real-time information about a Xen system and domains  
-v - output version information and exit  
-b - output data in batch mode  
-I - maximum number of iterations xentop should produce before ending

- To Start a VM:  
# xe vm-start vm=<VM-NAME>
- To get information regarding the VM:  
# xe vm-list name-label=<VM-NAME>
- To shut down VM:  
# xe vm-shutdown vm=<VM-NAME>
- To perform force migration of VM:  
# xe vm-migrate vm=<VM-UUID> destination=<host-server>  
host=<destination-server>
- To suspend VM:  
# xe vm-suspend vm=<VM-NAME>

## 4.8 Apache

The Apache HTTP Server (httpd) was launched in 1995 and it has been the most popular web server on the Internet since April 1996. It provides a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards [14]. This software has been used as an application running inside virtual machine and observed the performance of this web server at different load condition of virtual machine.

Downloaded httpd package along with its dependencies using following command:

```
#yum install --downloadonly --downloadaddir=/root/packages httpd
```

Then packages are copied to each virtual machine via Secure Copy command scp and then installed using following command:

```
# rpm -ivh *
```

Apache server has been started with following command:

```
#service httpd start
```

## 4.9 Httpperf

Httpperf is a tool to measure web server performance. The most basic operation of httpperf is to generate a fixed number of HTTP GET requests and to measure how many replies (responses) came back from the server and at what rate the responses arrived [15]. This tool was used to obtain the response time of the apache web server running inside virtual machine which was installed on client test virtual machine with the following steps:

The following commands were used to install httpperf on the system:

```
#wget
http://ftp.tu.chemnitz.de/pub/linux/dag/redhat/el7/en/x86_64/rpmforge/RPMS/rp
mforge-release-0.5.3-1.el7.rf.x86_64.rpm

#rpm -Uvh rpmforge-release-0.5.3-1.el7.rf.x86_64.rpm

#yum install httpperf
```

The operation of httpperf can be controlled through several options. The following command was used from client machine to request root document from running virtual machine.

```
#httpperf --server 192.168.56.21 --port 80 --num-conns 10 --rate 1
```

The above command causes httpperf to create a connection to a host 192.168.56.21, sends a request for the root document, receives the reply, closes the connection, and then prints some performance statistics, where

- server – Specifies hostname or IP address of webserver
- port - Specifies the port number of web server listening for request from client, in the test case it was http port 80 on which apache webserver's default page is available.
- num-conns – Specifies the total number of connections to create.
- rate – Specifies the rate of connection request to server

## 4.10 Stress

Stress is a workload generator for POSIX system. It imposes a configurable amount of CPU, memory, I/O, and disk stress on the system. It is written in C, and is a free software licensed under the GPLv2 [16]. The stress package is downloaded from yum repository on client test machine which has internet access, copied the downloaded packages to each virtual machine and installed. The command used to download package is as below:

```
#yum install --downloadonly --downloadaddir=/root/packages stress
```

The package is copied to each virtual machine via Secure Copy command scp and then installed using following command:

```
#scp stress-1.0.4-4.el6.x86_64.rpm root@192.168.56.20:/root/
```

```
#rpm -ivh stress-1.0.4-4.el6.x86_64.rpm
```

The following command was used to generate workload on virtual machines:

```
#stress --cpu 4 --io 3 --vm 2 --vm-bytes 256M --timeout 3000s
```

The workload was imposed on virtual machines initially by specifying four CPU-bound processes, three I/O-bound processes, and two memory allocator processes for 3000 seconds as shown in the above command. Then increased number of CPU bound processes to increase the load on virtual machines to observe the response time of apache web server running inside virtual machine at different load average.

## 4.11 Implementation of Algorithms

### 4.11.1 Shell Script

A shell script is a scripting language which is popular in Unix/Linux operating system. It provides a command-line interpreter for Unix system and provides the features of general-purpose programming language, such as control-flow constructs, variables, loops, functions, arrays, subroutines and so on.

The shell translates the commands and sends them to the system. Most Linux distributions are shipped with many shells. Every shell has its own features, and some of them are very popular among developers today. Some of the popular shells are sh shell, bash shell, ksh shell and Csh shell [17].

#### **4.11.2 Functions**

The algorithms presented on methodology section of this dissertation work has been implemented on the shell script and the source code is provided on Appendix B. The script consists of different functions to perform different operation like calculating resource usage, scaling resources and migrating VM from one host to another host.

## CHAPTER 5

### EXPERIMENT AND OBSERVATION

The experiment consists of two different testing environments where first was in physical hardware and the second was in virtualized environment which are described in detail in the implementation section of this dissertation work. In case of physical environment, three identical XenServers were setup with configuration of 7<sup>th</sup> generation intel i3 processor and having 8GB RAM on two servers and one having 4GB of RAM. The three identical VMs are created and assigned 1GB of RAM and 10GB of hard disk and all of them have apache web server installed and running. The NFS server was created in hardware having configuration i3 6<sup>th</sup> generation intel processor, 4GB of RAM and 1 TB hard disk. The client test machine and XenCenter enabled system are also configured and all the machines are assigned static IP and connected to the network.

In case of virtualized environment, three identical XenServers were setup in a single machine having configuration of Intel i3 7<sup>th</sup> generation processor, 12GB of RAM and Windows10 as operating system where Virtualbox was installed and on top of that created three XenServers and assigned 2GB RAM on each node. The three identical VMs, one NFS server and one client VM are also created and each of them have been allocated 512MB of RAM and all of them including XenServers are in the same virtual network and each of them have static IP address assigned and are reachable from each other. All the three VM have apache web server installed and running.

To perform experiment on algorithm, distributed three virtual machines across three different Xenserver nodes, the first node contains two virtual machines, second node contains one virtual machine and the third node contains no virtual machines as shown in the block diagram of figure 4.1 and figure 4.2 of the implementation section of this dissertation work. The objective of the experiment is to scale CPU resources efficiently and to perform automated live migration of virtual machines in case there is no further resource scaling is possible to optimize resource provisioning there by reducing the response time of the application running inside virtual machines.

In both testing environment, httpperf ran from client machine to one of the VM requesting document root for apache web server running inside VM and observed the response (reply) time of it received by client machine via httpperf. The experiment was performed to obtain the response time of apache webserver running on a VM at different load conditions. The load generator tool stress was used



to generate load dynamically for the VMs. The load to be generated can be varied to the required amount by using stress load generator imposing varying load to the virtual machine. The response time of apache web server running on a VM at one-minute and five-minute load average varying load from 0 to 25 with interval of 5 were observed and noted. The response time was observed initially without implementing the algorithm and then implementing the algorithm.

## 5.1 Response Time in One-Minute Load Average from Physical Test Environment

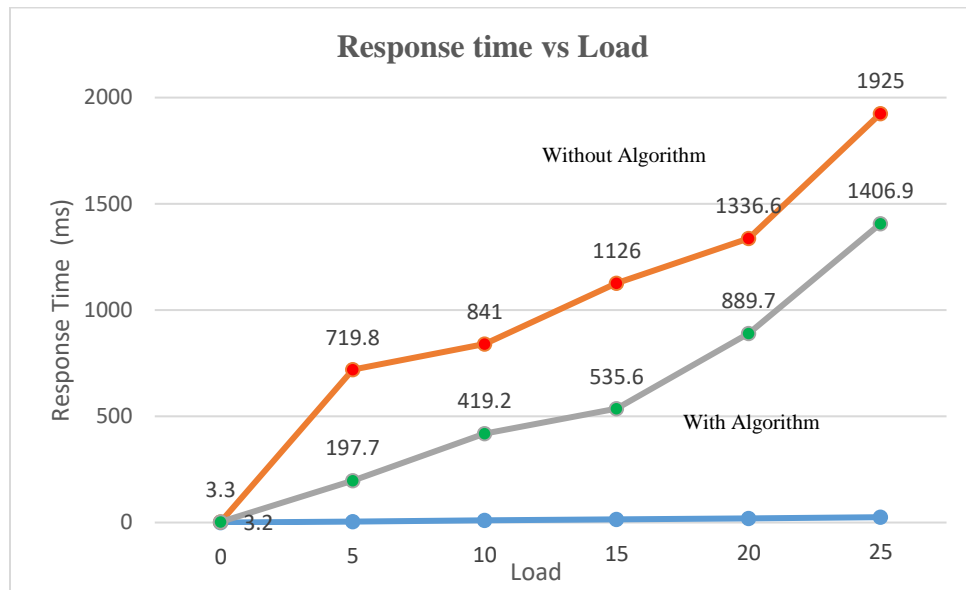


Figure 5.1: Response Time vs Load for One-Minute Load Average – Physical Test Environment

In one-minute load average, imposed load on a VM using stress tool and checked one-minute load average inside VM, once the load average reached the required state, httpperf was ran from client VM and observed the response time. Httpperf sends total of 10 requests to the web server with the speed of one request per second. As keeping stress imposing load on the VM, the load average increases and once the load reaches next state, httpperf was again ran and observed the response time at that state and continued till load average reaches 25 and respective response time were noted. The experiment was conducted two times, initially without algorithm implementation and next with algorithm implemented. From the data obtained, response time in millisecond (ms), it is

found that with algorithm implemented, response time of apache web server is better and relatively lower than without implementing the algorithm.

## 5.2 Response Time in Five-Minute Load Average from Physical Test Environment

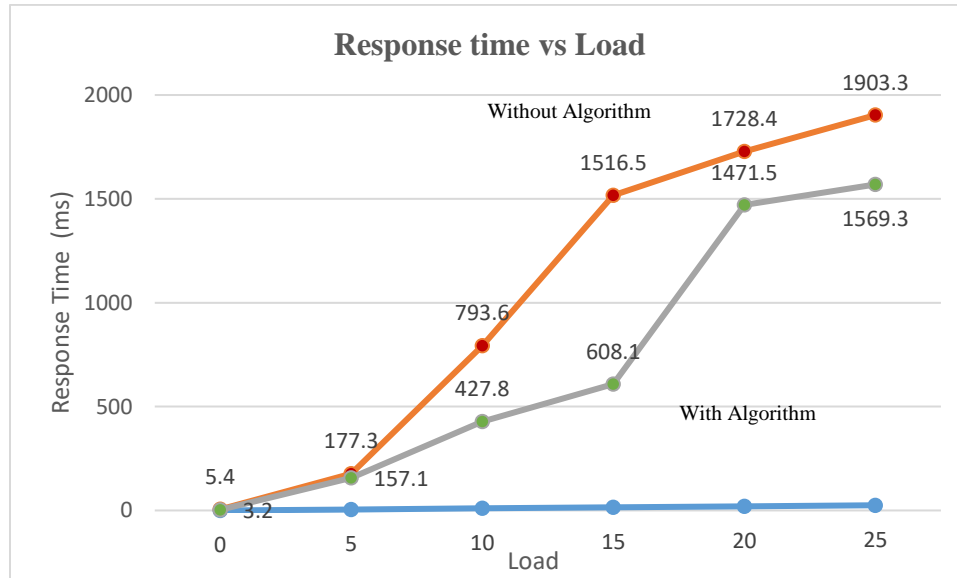


Figure 5.2: Response Time vs Load for Five-Minute Load Average – Physical Test Environment

In five-minute load average, load was imposed on a VM using stress and checked five-minute load average inside VM, once the load average reached the required state, httpperf was ran from client VM and observed the response time. Httpperf sends total of 10 requests to the web server with the speed of one request per second. As keeping stress imposing load on the VM, the load average increases and once the next interval of load average is reached, httpperf was ran again and observed the response time at that state and continued till load average reaches 25 and respective response time were noted. The experiment was conducted twice, initially without algorithm implementation and next with algorithm implemented. From the response time obtained which was in millisecond (ms), it is found that with the algorithm implemented, response time of apache web server is better and relatively lower than without implementing the algorithm.

### 5.3 Response Time in One-Minute Load Average from Virtualized Test Environment

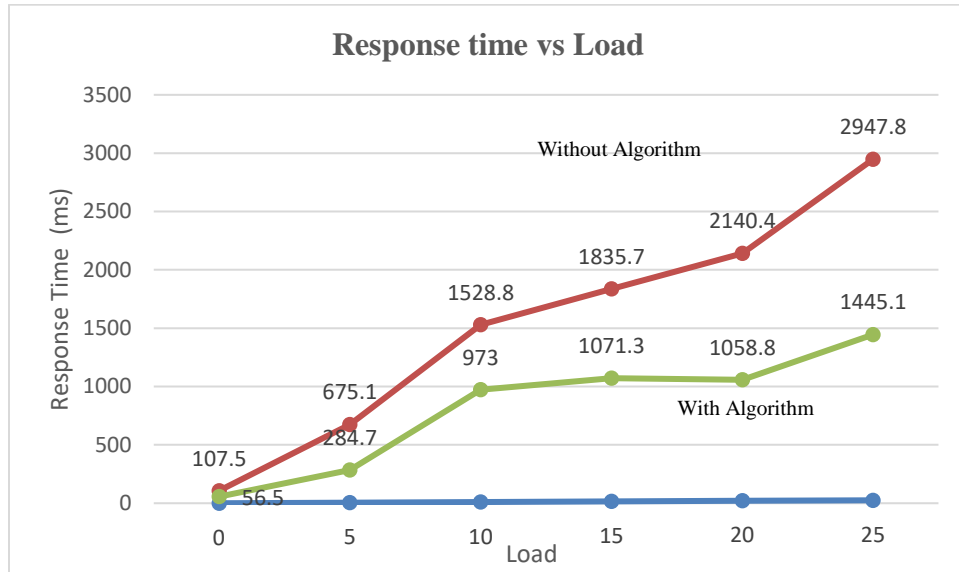


Figure 5.3: Response Time vs Load for One-Minute Load Average – Virtual Test Environment

In case of virtualized environment, the same experiment was repeated as in physical environment and noted the response time for one-minute load average. In the test, it is found that response time has the similar pattern as in case of physical test but with higher latency than the physical one and the graph between load vs response time is shown in figure 5.3 above.

## 5.4 Response Time in Five-Minute Load Average from Virtualized Test Environment

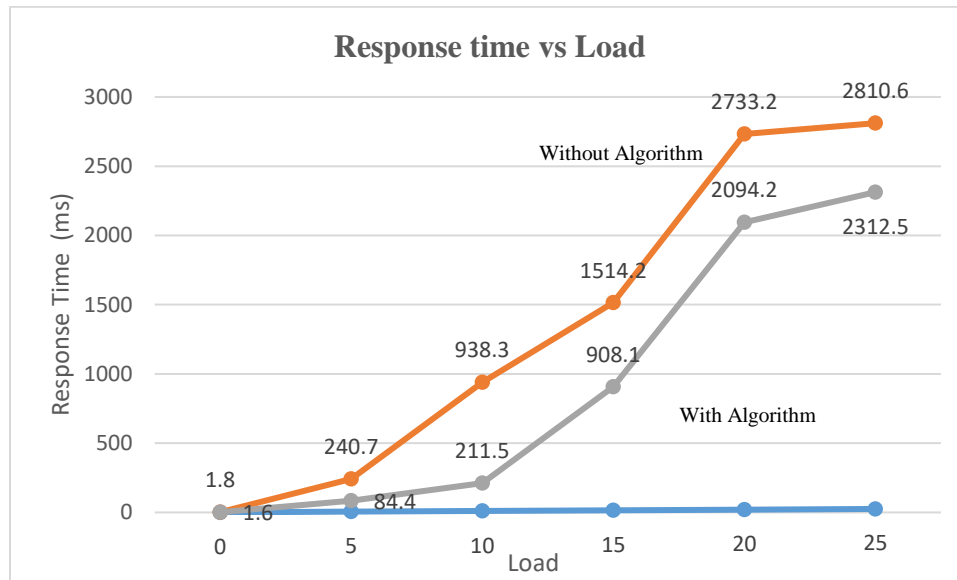


Figure 5.4: Response Time vs Load for Five-Minute Load Average – Virtual Test Environment

In case of virtualized environment, the same experiment was performed as in physical environment and noted the response time for five-minute load average. In the test, it is found that response time has the similar pattern as in case of physical test but with higher latency than the physical one and the graph between load vs response time is shown in figure 5.4 above.

## 5.5 Total Migration Time Calculation

The total migration time of virtual machines was calculated while it was being live migrated from one Xenserver host to another Xenserver host. The migration time was calculated with and without load conditions of VM with two different RAM size allocated to virtual machine in both physical and virtualized environments.

Table 5.1: Total Migration Time Calculation of VM live migration in Physical Test Environment

S.N.	Virtual Machine (OS)	RAM Size (MB)	Total Migration Time	Total Migration Time
			Without Load	With Load
1	CentOS 6.9	512	51 Sec	68 Sec
2	CentOS 6.9	1024	97 Sec	242 Sec

Table 5.2: Total Migration Time Calculation of VM live migration in Virtual Test Environment

S.N.	Virtual Machine (OS)	RAM Size (MB)	Total Migration Time	Total Migration Time
			Without Load	With Load
1	CentOS 6.9	512	13 Sec	20 Sec
2	CentOS 6.9	1024	17 Sec	28 Sec

From the results obtained from both test environment, it is observed that total migration time in case of virtual machine hosted in XenServer in virtual environment is less than compared to virtual machine hosted in physical XenServer. It is because all three XenServers (source and destination host) were in the same hardware machine hosted in VirtualBox and all of them are connected virtually in virtual test environment which has better speed than physical network.

## 5.6 Evaluation

From the experiment conducted in both cases physical and virtualized environment, it is found that the performance of the virtual machines is dependent on the amount of resources allocated to that particular VM. The response time of the virtual machines has been found increasing while increasing the load by stress load generator thereby decreasing the overall performance of the virtual machine. Also, it is noticed that background process running inside VM also affecting the performance of Virtual Machine. But with the algorithm implemented, the response time was comparatively better than without algorithm implementation.

Similarly, during the live migration of VM in both cases physical and virtualized environment, it is noticed that the complete down time or timeout of VM was only a single packet drop while doing continuous ping to VM being live migrated. But the total migration time was dependent on

the size of RAM allocated and load generated to the VM. It took longer time to migrate VM having 1GB RAM than having 512 MB. Also, it took longer time to migrate VM having high load than VM with low load. But while comparing the results between physical and virtual environment test, it is found that total migration time of VM is better in case of VM being hosted in Virtual XenServer host and response time is better in case of VM hosted in physical XenServer host.

In summary, dynamic resource scaling and live migration algorithm have been successful in deciding the situations in which migration is inevitable and the situations in which scaling of the resources is imminent.

## **CHAPTER 6**

### **CONCLUSION AND FUTURE WORK**

#### **6.1 Conclusion**

In this dissertation work, dynamic resource scaling and live migration of Virtual Machines in a cloud architecture was implemented which consists of Xen Servers, VMs, NFS and a client test machine. The experiment was conducted by generating load using stress tool and measured the performance of web server which was installed and running on a VM hosted on Xen Server. Httpperf tool was installed on a client test machine which was used to measure the response time of web server running on a VM hosted on XenServer. Based on the experiments conducted, it is observed that implementing dynamic resource scaling and live migration in the cloud has considerably improved the performance of the virtual machine in terms of response time.

It has been successful to perform automated live migration of virtual machine from one XenServer to another XenServer in both physical and virtualized environments. The major problem faced while setting up the physical test environment was to fulfil the requirement of three identical physical servers with exactly same CPU specifications for XenServes to meet the live migration criteria.

#### **6.2 Future Work**

For the future work, as live migration is the recent and most important topic in cloud computing, an important future work is to perform live migration of VM in cloud computing without identical CPU architecture and without shared storage which will enable the live migration in heterogeneous network architecture in cloud data centre.

## References

- [1] M P Gilesh, Sanjay Satheesh, Athul Chandran, S.D. Madhu Kumar, Lillykutty Jacob, "Parallel Schedule of Live Migrations for Virtual Machine Placements," 2018 IEEE 4<sup>th</sup> International Conference on Collaboration and Internet Computing
- [2] Gulshan Soni, Mala Kalra, "Comparative Study of Live Virtual Machine Migration Techniques in Cloud," International Journal of Computer Applications (0975-8887), Volume 84 - No 14, December 2013
- [3] Ahmed M. Mahfouz, Md Lutfar Rahman, Sajjan G. Shiva, "Secure Live Virtual Machine Migration through Runtime Monitors," Proceeding of 2017 Tenth International Conference on Contemporary Computing (IC3), 10-12 August, Noida, India
- [4] Anupam Tamrakar, "Security in Live Migration of Virtual Machine with Automated Load Balancing," International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 3, Issue 12, December-2014
- [5] Achar R., Thilagam P. S., Soans N., Vikyath P. V., Rao S., A.M. V., (2013). "Load Balancing in Cloud Based on Live Migration of Virtual Machines", 2013 Annual IEEE India Conference (INDICON), IEEE
- [6] Yi, Z. and Wenlong, H. (2009). "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud." Proceedings of Fifth International Joint Conference on INC, IMS and IDC, pp. 170-175, IEEE.
- [7] Jinhua, H., Jianhua, G., Guofei, S. and Tianhai, Z. (2010). "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment." Proceedings of International Symposium on Parallel Architectures, Algorithms and Programming, pp. 89-96, IEEE.
- [8] Xiaona, R., Rongheng, L. and Hua, Z. (2011). "A Dynamic Load Balancing Strategy for cloud computing platform based on exponential smoothing forecast." Proceedings of International Conference on Cloud Computing and Intelligence Systems, pp. 220-224, IEEE
- [9] Wenhong, T., Yong, Z., Yuanliang, Z., Minxian, X. and Chen, J. (2011). "A Dynamic and Integrated Loadbalancing Scheduling Algorithm for Cloud Datacenters." Proceedings of International Conference on Cloud Computing and Intelligence Systems, pp. 311-315, IEEE.

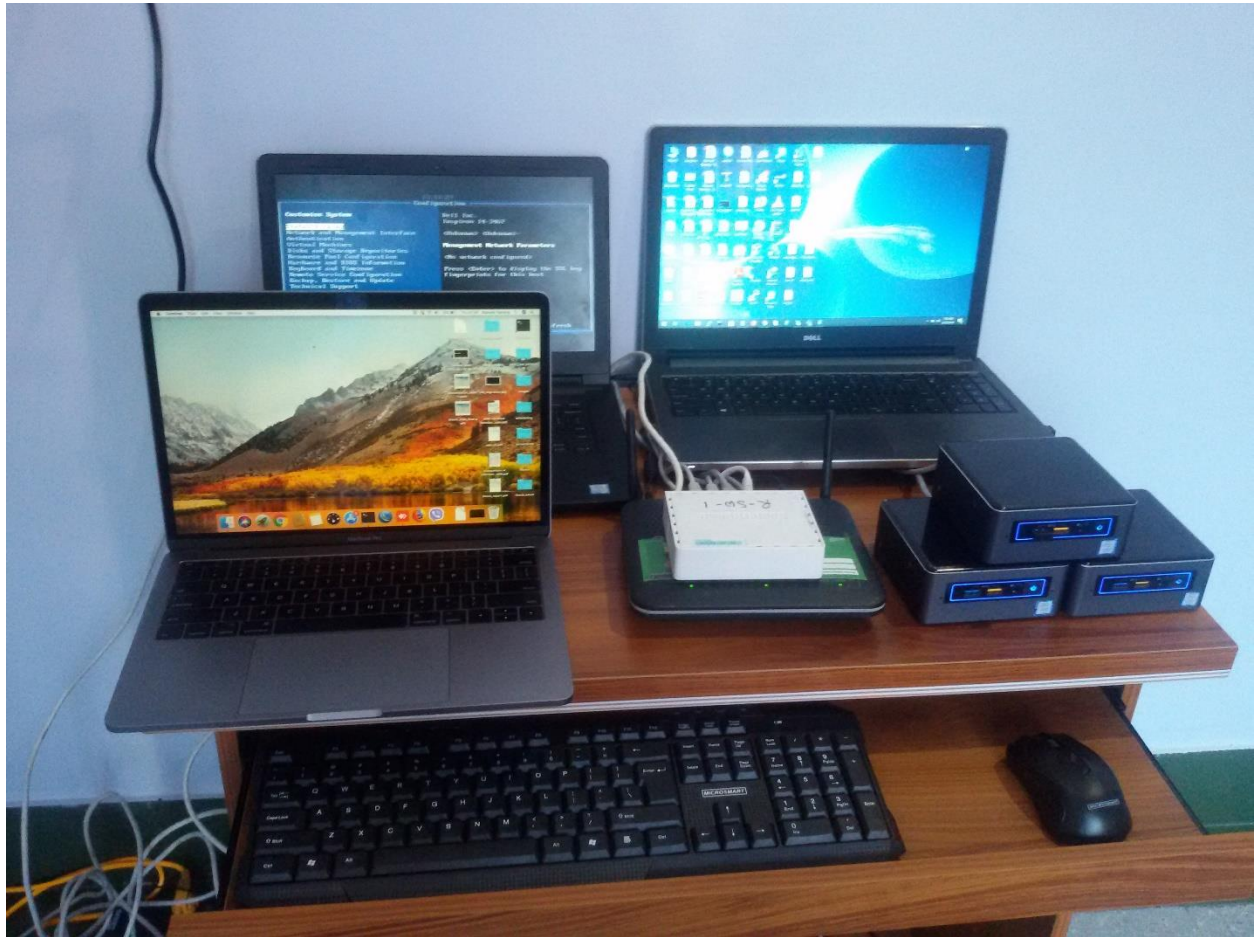


- [10] Gaochao, X., Junjie P. and Xiaodong, F.(2013).“ Load Balancing Model Based on Cloud Partitioning for the Public Cloud.” Proceedings of Tsinghua Science and Technology, pp. 34-39, IEEE
- [11] Haozheng, R., Yihua L.,Chao Y.(2012) “The Load Balancing Algorithm in Cloud Computing Environment.” International Conference on Computer Science and Network Technology, pp. 925-928, IEEE.
- [12] Jun Wu, Chen-Yuan Wang and Jian-Fu Li, "LA-Credit: A Load-Awareness Scheduling Algorithm for Xen Virtualized Platforms", 2016 IEEE 2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, IEEE International Conference on Intelligent Data and Security
- [13] Rajendra H. Bele, Dr. Chitra G.Desai, “Optimization of Default Credit Scheduler of Xen Virtualization Technology”, IICMR Research Journal I4, Vol.11-0 Issue1, December 2016, ISSN No.0975 2757, SJIF 2015: 4.27
- [14] “Apache,” [Online]. Available: <https://httpd.apache.org/> [Accessed 26 05 2019].
- [15] “Httpperf,” [Online]. Available: <https://linux.die.net/man/1/httpperf> [Accessed 26 05 2019]
- [16] “Stress,” [Online]. Available: <https://people.seas.harvard.edu/~apw/stress/> [Accessed 26 05 2019]
- [17] “VirtualBox,” [Online]. Available: <https://www.virtualbox.org/wiki/VirtualBox> [Accessed 26 05 2019]
- [18] Ebrahim Mokhtar, Mallett Andrew, Mastering Linux Shell Scripting: A practical guide to Linux command-line, Bash scripting, and Shell programming (Packt Publishing, Second Edition, 2018 April, Birmingham, UK).

## Appendix A

### Screen Shots

1. Picture taken during XenServer Cloud Environment setup in Physical hardware



## 2. XenCenter GUI showing XenServer and VM info

The screenshot displays the XenCenter application window. The left sidebar shows a tree view of the infrastructure, including Pools, Servers, VMs, and various storage repositories. The main pane, titled 'Objects by Type', shows a table of resources with columns for Name, CPU Usage, Used Memory, Disks, Network, Address, and Uptime. The table lists three servers (xenph2, xenph1, xenph3) and five VMs (vm1, vm2, vm3, vm4, vm5).

Type	Name	CPU Usage	Used Memory	Disks (avg / max KBs)	Network (avg / max KBs)	Address	Uptime
Pools	xen-physical-test XenServer Physical Machine Test	-	-	-	-	-	-
Servers	xenph2 Default install of XenServer	1% of 4 CPUs	1 GB of 7.9 GB	-	1/1	192.168.254.90	14 hours 13 minutes
Servers	xenph1 Default install of XenServer	1% of 4 CPUs	2 GB of 3.8 GB	-	1/1	192.168.254.91	1 hour 18 minutes
Servers	xenph3 Default install of XenServer	30% of 4 CPUs	5.1 GB of 7.9 GB	-	483/483	192.168.254.92	1 hour 22 minutes
VMs	vm1	100% of 1 CPU	149 MB of 1 GB	567195/1134390	0/0	fe80::109f:bfff:feb3::...	7 hours 4 minutes
VMs	vm2	0% of 1 CPU	175 MB of 1 GB	377633/755265	1/1	fe80::7ccd:56ff:fe20::...	7 hours 3 minutes
VMs	vm3	0% of 1 CPU	165 MB of 1 GB	540647/1081293	0/0	fe80::809c:aeff:fe8b::...	7 hours 4 minutes
VMs	vm4	4% of 1 CPU	165 MB of 1 GB	12/24	0/0	fe80::b8d6:92ff:fe94::...	7 hours 3 minutes
VMs	vm5	0% of 1 CPU	175 MB of 1 GB	5/10	0/0	fe80::898:14ff:fedc:6::...	7 hours 3 minutes

### 3. Xen commands in action showing XenServer and VM info

```
~ — root@xenph2:~ — ssh root@192.168.254.90
[root@xenph2 ~]# xe host-list
uuid ( RO)           : 8caaeece7-65cf-4850-a067-0c29ba9976e4
  name-label ( RW): xenph3
  name-description ( RW): Default install

uuid ( RO)           : cf3584e0-d5a1-49b3-af7c-8fdd1b2fd931
  name-label ( RW): xenph2
  name-description ( RW): Default install

uuid ( RO)           : 3a731886-f4dd-438b-8214-62d7416e3c07
  name-label ( RW): xenph1
  name-description ( RW): Default install

[root@xenph2 ~]# xe vm-list
uuid ( RO)           : 05162498-5237-1757-b133-b74856a4aa9f
  name-label ( RW): vm3
  power-state ( RO): running

uuid ( RO)           : c74626b1-8b2c-53d3-2a32-7a6f978f9159
  name-label ( RW): vm4
  power-state ( RO): running

uuid ( RO)           : a5ccc592-6fa8-4167-9f69-75c2fc1e427e
  name-label ( RW): Control domain on host: xenph1
  power-state ( RO): running

uuid ( RO)           : d74d9ea8-6ce8-2370-b9c0-59f594574bf7
  name-label ( RW): vm1
  power-state ( RO): running

uuid ( RO)           : 8e85a96d-e7cd-41f4-8d1e-324a79069750
  name-label ( RW): Control domain on host: xenph3
  power-state ( RO): running

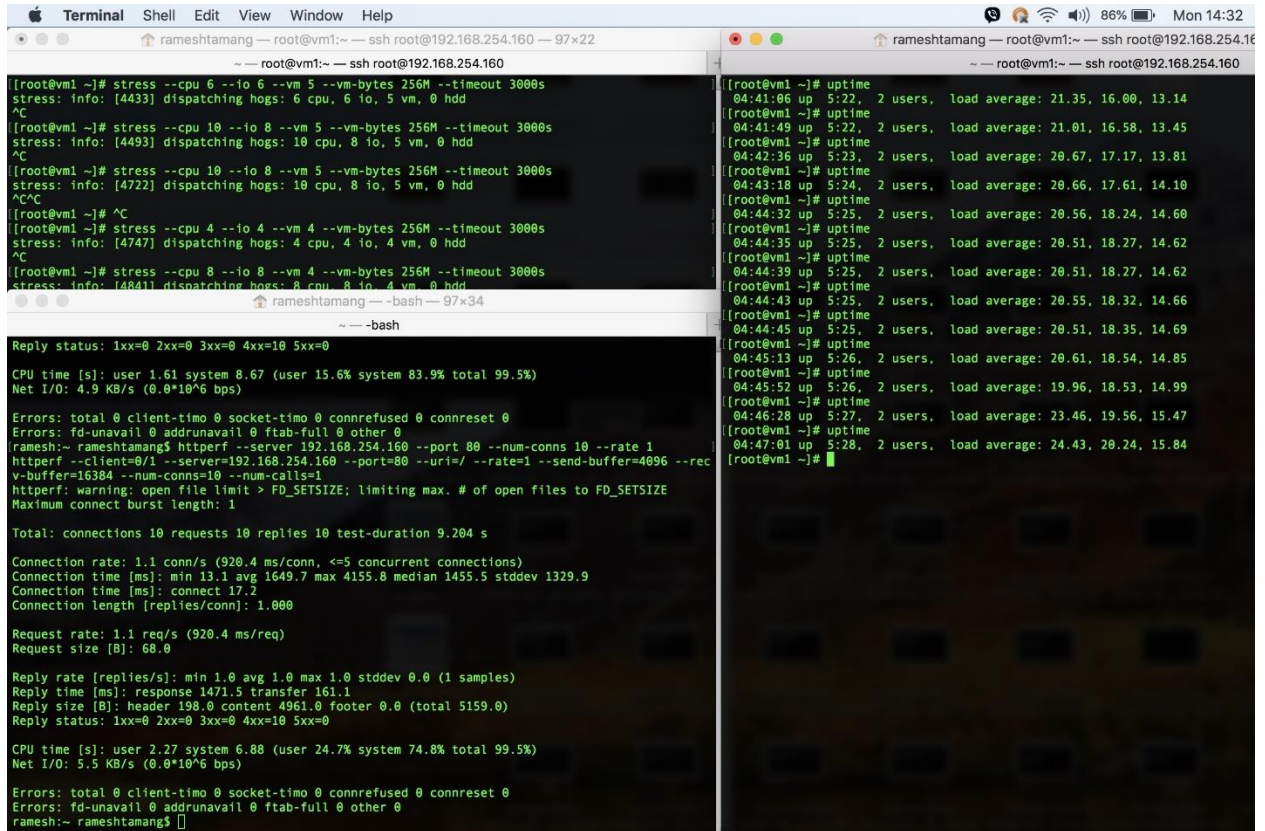
uuid ( RO)           : 1ffd9631-753f-44cd-b5c4-5b0bb1b7788f
  name-label ( RW): Control domain on host: xenph2
  power-state ( RO): running

uuid ( RO)           : 2a4ecee1-c29b-8d16-8163-f58ebdb2f22
  name-label ( RW): vm5
  power-state ( RO): running

uuid ( RO)           : 2a3a711f-0d2f-cbd5-0791-ebe14737f477
  name-label ( RW): vm2
  power-state ( RO): running

[root@xenph2 ~]#
```

#### 4. Stress is generating load on VM, Checking Load Average and Running httpperf in Physical Environment



```
Terminal Shell Edit View Window Help
rameshtamang ~ root@vm1:~ ssh root@192.168.254.160 97x22
~ root@vm1:~ ssh root@192.168.254.160

[root@vm1 ~]# stress --cpu 6 --io 6 --vm 5 --vm-bytes 256M --timeout 3000s
stress: info: [4433] dispatching hogs: 6 cpu, 6 io, 5 vm, 0 hdd
^C
[root@vm1 ~]# stress --cpu 10 --io 8 --vm 5 --vm-bytes 256M --timeout 3000s
stress: info: [4493] dispatching hogs: 10 cpu, 8 io, 5 vm, 0 hdd
^C
[root@vm1 ~]# stress --cpu 10 --io 8 --vm 5 --vm-bytes 256M --timeout 3000s
stress: info: [4722] dispatching hogs: 10 cpu, 8 io, 5 vm, 0 hdd
^C
[root@vm1 ~]# ^C
[root@vm1 ~]# stress --cpu 4 --io 4 --vm 4 --vm-bytes 256M --timeout 3000s
stress: info: [4747] dispatching hogs: 4 cpu, 4 io, 4 vm, 0 hdd
^C
[root@vm1 ~]# stress --cpu 8 --io 8 --vm 4 --vm-bytes 256M --timeout 3000s
stress: info: [4841] dispatching hogs: 8 cpu, 8 io, 4 vm, 0 hdd
^C
rameshtamang ~ bash 97x34
~ bash

Reply status: 1xx=0 2xx=0 3xx=0 4xx=10 5xx=0

CPU time [s]: user 1.61 system 0.67 (user 15.6% system 83.9% total 99.5%)
Net I/O: 4.9 KB/s (0.0*10^6 bps)

Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
rameshtamang$ httpperf --server 192.168.254.160 --port 80 --num-conns 10 --rate 1
httpperf --client=0/1 --server=192.168.254.160 --port=80 --uri=/ --rate=1 --send-buffer=4096 --rec
v-buffer=16384 --num-conns=10 --num-calls=1
httpperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to FD_SETSIZE
Maximum connect burst length: 1

Total: connections 10 requests 10 replies 10 test-duration 9.204 s

Connection rate: 1.1 conn/s (920.4 ms/conn, <=5 concurrent connections)
Connection time [ms]: min 13.1 avg 1649.7 max 4155.8 median 1455.5 stddev 1329.9
Connection time [ms]: connect 17.2
Connection length [replies/conn]: 1.000

Request rate: 1.1 req/s (920.4 ms/req)
Request size [B]: 68.0

Reply rate [replies/s]: min 1.0 avg 1.0 max 1.0 stddev 0.0 (1 samples)
Reply time [ms]: response 1471.5 transfer 161.1
Reply size [B]: header 198.0 content 4961.0 footer 0.0 (total 5159.0)
Reply status: 1xx=0 2xx=0 3xx=0 4xx=10 5xx=0

CPU time [s]: user 2.27 system 6.88 (user 24.7% system 74.8% total 99.5%)
Net I/O: 5.5 KB/s (0.0*10^6 bps)

Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
rameshtamang$
```

```
rameshtamang ~ root@vm1:~ ssh root@192.168.254.160
~ root@vm1:~ ssh root@192.168.254.160

[root@vm1 ~]# uptime
04:41:06 up 5:22, 2 users, load average: 21.35, 16.00, 13.14
[root@vm1 ~]# uptime
04:41:49 up 5:22, 2 users, load average: 21.01, 16.58, 13.45
[root@vm1 ~]# uptime
04:42:36 up 5:23, 2 users, load average: 20.67, 17.17, 13.81
[root@vm1 ~]# uptime
04:43:18 up 5:24, 2 users, load average: 20.66, 17.61, 14.10
[root@vm1 ~]# uptime
04:44:32 up 5:25, 2 users, load average: 20.56, 18.24, 14.60
[root@vm1 ~]# uptime
04:44:35 up 5:25, 2 users, load average: 20.51, 18.27, 14.62
[root@vm1 ~]# uptime
04:44:39 up 5:25, 2 users, load average: 20.51, 18.27, 14.62
[root@vm1 ~]# uptime
04:44:43 up 5:25, 2 users, load average: 20.55, 18.32, 14.66
[root@vm1 ~]# uptime
04:44:45 up 5:25, 2 users, load average: 20.51, 18.35, 14.69
[root@vm1 ~]# uptime
04:45:13 up 5:26, 2 users, load average: 20.61, 18.54, 14.85
[root@vm1 ~]# uptime
04:45:52 up 5:26, 2 users, load average: 19.96, 18.53, 14.99
[root@vm1 ~]# uptime
04:46:28 up 5:27, 2 users, load average: 23.46, 19.56, 15.47
[root@vm1 ~]# uptime
04:47:01 up 5:28, 2 users, load average: 24.43, 20.24, 15.84
[root@vm1 ~]#
```

## 5. Live Migration Script running in one of the Xenserver showing resource scaling and live migration in Physical Environment

```
~ — root@xenph2:~ — ssh root@192.168.254.90
[[root@xenph2 ~]# date
Mon Aug 19 09:05:28 NPT 2019
[[root@xenph2 ~]# ./livemigration.sh 2
Running Live Migration Script by root at 2019-08-19
XenServer xenph2 has id cf3584e0-d5a1-49b3-af7c-8fdd1b2fd931
Total number of VM running on xenph2 is 3
vm3
Cpupool Pool-0: tslice=30ms ratelimit=1000us
Name      ID Weight Cap
Domain-0  0     64   0
vm1       1    256   0
vm3       2    256  10
vm4       3    256   0
vm4
Cpupool Pool-0: tslice=30ms ratelimit=1000us
Name      ID Weight Cap
Domain-0  0     64   0
vm1       1    256   0
vm3       2    256  10
vm4       3    256  10
vm1
Cpupool Pool-0: tslice=30ms ratelimit=1000us
Name      ID Weight Cap
Domain-0  0     64   0
vm1       1    256  10
vm3       2    256  10
vm4       3    256  10
***** Candidate VM for resource scaling: vm3 *****
Comparing CPU usage 0
Scaling down CPU of vm3 by 20 %
Scaling down CPU of vm3 by 20 %
Migrating VM...
vm3 has been migrated from xenph2 to xenph3 successfully.
Number of VM running on xenph2 is 2
***** Candidate VM for resource scaling: vm4 *****
Comparing CPU usage 0
Scaling down CPU of vm4 by 20 %
Scaling down CPU of vm4 by 20 %
Migrating VM...
```



## 6. Stress is generating load on VM, Checking Load Average and Running httpperf in Virtualized Environment

```
root@vm3:~  
[root@vm3 ~]# stress --cpu 12 --io 3 --vm 2 --vm-bytes 256M --timeout 3000s  
stress: info: [2470] dispatching hogs: 12 cpu, 3 io, 2 vm, 0 hdd  
^C  
[root@vm3 ~]# stress --cpu 16 --io 3 --vm 2 --vm-bytes 256M --timeout 3000s  
stress: info: [2508] dispatching hogs: 16 cpu, 3 io, 2 vm, 0 hdd  
^C  
[root@vm3 ~]# stress --cpu 18 --io 3 --vm 2 --vm-bytes 256M --timeout 3000s  
stress: info: [2577] dispatching hogs: 18 cpu, 3 io, 2 vm, 0 hdd  
^C  
[root@vm3 ~]# stress --cpu 18 --io 3 --vm 2 --vm-bytes 256M --timeout 3000s  
stress: info: [2661] dispatching hogs: 18 cpu, 3 io, 2 vm, 0 hdd  
^C  
[root@vm3 ~]#  
  
root@client:~  
httpperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to  
FD_SETSIZE  
Maximum connect burst length: 1  
  
Total: connections 10 requests 10 replies 10 test-duration 11.085 s  
  
Connection rate: 0.9 conn/s (1108.5 ms/conn, <=5 concurrent connections)  
Connection time [ms]: min 83.8 avg 3033.7 max 7084.2 median 2822.5 stddev 2099.8  
Connection time [ms]: connect 85.8  
Connection length [replies/conn]: 1.000  
  
Request rate: 0.9 req/s (1108.5 ms/req)  
Request size [B]: 66.0  
  
Reply rate [replies/s]: min 0.4 avg 0.9 max 1.4 stddev 0.7 (2 samples)  
Reply time [ms]: response 2947.8 transfer 0.1  
Reply size [B]: header 198.0 content 4961.0 footer 0.0 (total 5159.0)  
Reply status: lxx=0 2xx=0 3xx=0 4xx=10 5xx=0  
  
CPU time [s]: user 1.39 system 9.69 (user 12.6% system 87.4% total 100.0%)  
Net I/O: 4.6 KB/s (0.0*10^6 bps)  
  
root@vm3:~  
[root@vm3 ~]# uptime  
03:21:36 up 2:49, 2 users, load average: 24.95, 18.75, 13.11  
[root@vm3 ~]# uptime  
03:21:38 up 2:49, 2 users, load average: 24.95, 18.85, 13.17  
[root@vm3 ~]# uptime  
03:21:39 up 2:49, 2 users, load average: 24.95, 18.85, 13.17  
[root@vm3 ~]# uptime  
03:21:41 up 2:49, 2 users, load average: 24.95, 18.85, 13.17  
[root@vm3 ~]# uptime  
03:21:42 up 2:49, 2 users, load average: 24.87, 18.94, 13.23  
[root@vm3 ~]# uptime  
03:21:45 up 2:49, 2 users, load average: 24.87, 18.94, 13.23  
[root@vm3 ~]# uptime  
03:21:51 up 2:49, 2 users, load average: 24.80, 19.02, 13.29  
[root@vm3 ~]# uptime  
03:21:53 up 2:49, 2 users, load average: 24.90, 19.14, 13.36  
[root@vm3 ~]# uptime  
03:21:54 up 2:49, 2 users, load average: 24.90, 19.14, 13.36  
[root@vm3 ~]# uptime  
03:21:55 up 2:49, 2 users, load average: 24.90, 19.14, 13.36  
[root@vm3 ~]# uptime  
03:21:58 up 2:49, 2 users, load average: 25.15, 19.28, 13.43  
[root@vm3 ~]# uptime  
03:23:07 up 2:51, 2 users, load average: 24.22, 20.23, 14.19  
[root@vm3 ~]#
```

## 7. Live Migration Script running in one of the Xenserver showing resource scaling and live migration in virtualized environment

```
root@xennode8:~  
[root@xennode8 ~]# ./live_migration.sh 8  
Running Live Migration Script by root at 2019-04-29  
XenServer xennode8 has id dd5f4250-fcb4-48cd-b87b-963e95768709  
Total number of VM running on xennode8 is 2  
vml  
Cpupool Pool-0: tslice=30ms ratelimit=1000us  
Name ID Weight Cap  
Domain-0 0 256 0  
vml 24 256 10  
vm3 25 256 0  
vm3  
Cpupool Pool-0: tslice=30ms ratelimit=1000us  
Name ID Weight Cap  
Domain-0 0 256 0  
vml 24 256 10  
vm3 25 256 10  
***** Candidate VM for resource scaling: vml *****  
Comparing CPU usage 90  
Scaling up CPU of vml by 10 %  
Scaling Complete  
Scaling up CPU of vml by 10 %  
Scaling Complete  
Migrating VM...  
vml has been migrated from xennode8 to xennode7 successfully.  
Number of VM running on xennode8 is 1  
***** Candidate VM for resource scaling: vm3 *****  
Comparing CPU usage 90  
Scaling up CPU of vm3 by 10 %  
Scaling Complete  
Scaling up CPU of vm3 by 10 %  
Scaling Complete  
Migrating VM...  
vm3 has been migrated from xennode8 to xennode9 successfully.  
No running VM found on this server  
Server is ready to accept VMs  
No running VM found on this server  
Server is ready to accept VMs  
^C  
[root@xennode8 ~]#
```



## 8. Required software (Apache and stress) installation in one of the virtual machine

```
root@vm4:~/packages
[root@vm4 stress-tools]# ls
httpperf-0.9.0-1.el6.rf.x86_64.rpm stress-1.0.4-4.el6.x86_64.rpm
[root@vm4 stress-tools]# rpm -ivh httpperf-0.9.0-1.el6.rf.x86_64.rpm stress-1.0.4-4.el6.x86_64.rpm
warning: httpperf-0.9.0-1.el6.rf.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 6b8d79e6: NOKEY
warning: stress-1.0.4-4.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY
Preparing... ##### [100%]
 1:stress ##### [ 50%]
 2:httpperf ##### [100%]
[root@vm4 stress-tools]# cd ..
[root@vm4 ~]# cd packages/
[root@vm4 packages]# ls
apr-1.3.9-5.el6_9.1.x86_64.rpm apr-util-ldap-1.3.9-3.el6_0.1.x86_64.rpm httpd-tools-2.2.15-69.el6.centos.x86_64.rpm
apr-util-1.3.9-3.el6_0.1.x86_64.rpm httpd-2.2.15-69.el6.centos.x86_64.rpm mailcap-2.1.31-2.el6.noarch.rpm
[root@vm4 packages]# rpm -ivh *
warning: apr-1.3.9-5.el6_9.1.x86_64.rpm: Header V3 RSA/SHA1 Signature, key ID c105b9de: NOKEY
Preparing... ##### [100%]
 1:apr ##### [ 17%]
 2:apr-util ##### [ 33%]
 3:apr-util-ldap ##### [ 50%]
 4:httpd-tools ##### [ 67%]
 5:mailcap ##### [ 83%]
 6:httpd ##### [100%]
[root@vm4 packages]#
```

## 9. Shared storage mounted to XenServer

```
root@xennode8:~
[root@xennode8 ~]# df -h

```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdal	18G	1.8G	16G	11%	/
devtmpfs	289M	0	289M	0%	/dev
tmpfs	299M	48K	299M	1%	/dev/shm
tmpfs	299M	780K	298M	1%	/run
tmpfs	299M	0	299M	0%	/sys/fs/cgroup
xenstore	299M	0	299M	0%	/var/lib/xenstored
/dev/loop0	55M	55M	0	100%	/var/xen/xo-install
/dev/sda5	4.0G	182M	3.6G	5%	/var/log
192.168.56.102:/nfsfileshare/dfea6a41-d0a2-66e8-61de-47221274bb7d	39G	21G	19G	52%	/run/sr-mount/dfea6a41-d0a2-66e8-61de-47221274bb7d
192.168.56.102:/iso/83ac4e37-bbc2-e5f6-ba9a-d22bb035ff08	39G	21G	19G	52%	/run/sr-mount/83ac4e37-bbc2-e5f6-ba9a-d22bb035ff08
192.168.56.102:/iso	39G	21G	19G	52%	/run/sr-mount/dfdd1287-01db-eacb-6f47-81097a5c8a3b
tmpfs	60M	0	60M	0%	/run/user/0

```
[root@xennode8 ~]#
```

## Appendix B

### Source Code

```
#!/usr/bin/sh

#set -xv    #Set while debugging

current_date=`date +%Y-%m-%d`

echo "Running Live Migration Script by $USER at $current_date"

if [ -z "$1" ]
then
    echo "ERROR: Argument Missing: Please run script as ./livemigration.sh <XenServer ID>"
    echo "For Example: ./livemigration.sh 8"
    exit 1
else
    hostnode_ID=$1
    host_node="xennode$1"
fi

# Variable Initialization

rm -f /root/xen_logs/*

count=0

cpu=0

i=0

init_cpu=0

init_ram=0

no_nodes=8

#candidate_vm=0

vm_uuid=0

host_UUID=$(xe host-list name-label=$host_node |grep -a uuid|cut -c 29-100)

echo "XenServer $host_node has id $host_UUID"

num_VM=$(xe vm-list resident-on=$host_UUID power-state=running|grep name|wc -l|cut -c 1-2)

# Need to subtract count of Xenhostnode to get exact number of VM with power state running

num_VM=$((num_VM - 1))

echo "Total number of VM running on $host_node is $num_VM"
```

```

running_VM=$(xe vm-list resident-on=$host_UUID power-state=running|grep vm|head -n $num_VM|cut
-c 24-100)

echo "$running_VM" >> /root/xen_logs/vm_name.txt

while read vm_reschedule
do

    xl sched-credit -d $vm_reschedule -c 10

    echo $vm_reschedule

    xl sched-credit

done < /root/xen_logs/vm_name.txt

# Function takes 4 arguments: 1: Scaling amount in percentage 2: Scaling Rule 3: VM Name 4:
RAM or CPU Resource to scale

function Scale_Resource()
{
    case $4 in
    1)
        if [ $2 -eq 1 ]
        then
            echo "Scaling up CPU of $3 by $1 % "
            scale_pct=$(xl sched-credit|grep -a $3|cut -c 48-50)
            if [ $scale_pct -lt 100 ]
            then
                cpu_cap1=$((scale_pct * $1))
                cpu_cap2=$((cpu_cap1 / 100))
                recap=$((scale_pct + cpu_cap2))
                if [ $recap -lt 100 ]
                then
                    xl sched-credit -d $3 -c $recap
                    echo "Scaling Complete"
                else
                    xl sched-credit -d $3 -c 100
                    echo "Reached maximum hardware performance of Xen Server, further scaling is not
possible"
                fi
            fi
        fi
    fi
}

```

```

        fi
    fi
    if [ $2 -eq 0 ]
    then
        echo "Scaling down CPU of $3 by $1 % "
        scale_pct=$(xl sched-credit|grep -a $3|cut -c 48-50)
        if [ $scale_pct -gt 20 ]
        then
            cpu_ucap1=$((scale_pct * $1))
            cpu_ucap2=$((cpu_ucap1 / 100))
            recap=$((scale_pct - $cpu_ucap2))
            xl sched-credit -d $3 -c $recap
            echo "Scaling Complete"
            return 1
        fi
    fi
;;
2)
vm_uuid=$(xe vm-list name-label=$candidate_vm power-state=running|grep uuid|cut -c 24-59)
static_mem_min=$(xe vm-param-get uuid=$vm_uuid param-name=memory-static-min)
static_mem_max=$(xe vm-param-get uuid=$vm_uuid param-name=memory-static-max)
dynamic_mem_min=$(xe vm-param-get uuid=$vm_uuid param-name=memory-dynamic-min)
dynamic_mem_max=$(xe vm-param-get uuid=$vm_uuid param-name=memory-dynamic-max)
if [ $2 -eq 0 ]
then
    echo "Scaling down RAM of $3 by $1 % "
    if [ $dynamic_mem_max -lt $static_mem_min ]
    then
        pc=$(( $1 / 100 ))
        rt=$((dynamic_mem_max * $pc))
        dynamic_static_max=$((dynamic_mem_max + $rt))
        xe vm-param-set uuid=$vm_uuid memory-dynamic-max=$dynamic_static_max
    fi
fi

```

```

        fi
    fi
    if [ $2 -eq 1 ]
    then
        echo "Scaling up RAM of $3 by $1 %"
        if [ $dynamic_mem_max -lt $static_mem_min ]
        then
            pc=$(( $1 / 100 ))
            rt=$(( $dynamic_mem_max * $pc ))
            dynamic_static_max=$(( $dynamic_mem_max + $rt ))
            xe vm-param-set uuid=$vm_uuid memory-dynamic-max=$dynamic_static_max
        fi
    fi
    ;;
esac
}

# Function to migrate VM to suitable XenServer Host
function MigrateToServer()
{
    inc_id=$(( $hostnode_ID + 1 ))
    dec_id=$(( $hostnode_ID - 1 ))

    if [ $inc_id -gt $no_nodes ] && [ $dec_id -lt $no_nodes ] && [ $inc_id -gt 0 ] && [ $dec_id -gt 0 ]
    then
        # Find Available Free Memory of Suitable XenServers
        free_mem_inc_id=$(xe host-list name-label=xennode$inc_id params=memory-free --minimal)
        free_mem_dec_id=$(xe host-list name-label=xennode$dec_id params=memory-free --minimal)
        vm_uuid=$(xe vm-list name-label=$candidate_vm power-state=running|grep uuid|cut -c 24-59)
    fi

    echo "Migrating VM..."

#Compare Free Memory Available
    if [ $free_mem_inc_id -gt $free_mem_dec_id ]

```

```

then

xe vm-migrate vm=$vm_uuid destination="$host_node" host="xennode$inc_id" --live
echo "$candidate_vm has been migrated from $host_node to xennode$inc_id successfully."
else

xe vm-migrate vm=$vm_uuid destination="$host_node" host="xennode$dec_id" --live
echo "$candidate_vm has been migrated from $host_node to xennode$dec_id successfully."

fi
}

# Function to calculate CPU utilization
function Calc_CPU_usage()
{
cpu=$(xentop -v -b -i 2|grep $candidate_vm|cut -c 32-35)
echo "$cpu" >> /root/xen_logs/cpu_val$i.txt
act=$(cut -c 1-2 /root/xen_logs/cpu_val$i.txt|head -n 2|tail -n 1)
sc=$(xl sched-credit|grep -a $candidate_vm|cut -c 48-50)
if [ -z "$sc" ]; then
# if value of variable $sc is zero, i.e no running VM, need to exit from loop
break
else
pac=$((act * 100))
init_cpu=$(( pac / sc ))
fi
}

function Calc_RAM_usage()
{
ram="$(xentop -v -b -i 1|grep $candidate_vm|cut -c 50-54)"
echo "$ram" >> /root/xen_logs/ram_val$i.txt
init_ram=$(cut -c 1-2 /root/xen_logs/ram_val$i.txt|head -n 1)
}

# Main function starts from here
### Run script in infinite loop ###
while [ $count -lt 2 ]

```

```

do
num_VM=$(xe vm-list resident-on=$host_UUID power-state=running|grep name|wc -l|cut -c 1-2)
# Need to subtract count of Xenhostnode to get exact number of VM with power state running
num_VM=$(( $num_VM - 1 ))
if [ $num_VM -gt 0 ]
then
while read candidate_vm
do
temp=$(xl sched-credit|grep -a $candidate_vm|cut -c 48-50)
if [ $temp -eq 0 ]
then
xl sched-credit -d $candidate_vm -c 10
echo "VM CAP value modified"
fi
echo "***** Candidate VM for resource scaling: $candidate_vm *****"
Calc_CPU_usage
echo "Comparing CPU usage $init_cpu"
if [ "$init_cpu" -gt 70 ] && [ "$init_cpu" -lt 95 ];
then
Scale_Resource 10 1 $candidate_vm 1
sleep 3
Calc_CPU_usage
if [ "$init_cpu" -gt 70 ] && [ "$init_cpu" -lt 95 ];
then
Scale_Resource 10 1 $candidate_vm 1
sleep 3
Calc_CPU_usage
fi
if [ "$init_cpu" -gt 70 ] && [ "$init_cpu" -lt 95 ];
then
MigrateToServer
fi

```

```

fi
Calc_CPU_usage
if [ ! -z "$init_cpu" ] && [ "$init_cpu" -gt 95 ];
then
    Scale_Resource 20 1 $candidate_vm 1
    sleep 3
    Calc_CPU_usage
    if [ ! -z "$init_cpu" ] && [ "$init_cpu" -gt 95 ];
    then
        MigrateToServer
    fi
fi
Calc_CPU_usage
if [ ! -z "$init_cpu" ] && [ "$init_cpu" -le 10 ]
then
    Scale_Resource 20 0 $candidate_vm 1
    sleep 3
    Calc_CPU_usage
    if [ ! -z "$init_cpu" ] && [ "$init_cpu" -le 10 ]
    then
        Scale_Resource 20 0 $candidate_vm 1
        sleep 3
        Calc_CPU_usage
        if [ ! -z "$init_cpu" ] && [ "$init_cpu" -le 10 ]
        then
            MigrateToServer
        fi
    fi
fi
Calc_RAM_usage
if [ -z "$init_ram" ]; then
break

```



```

echo "Comparing RAM usage $init_ram"
if [ "$init_ram" -gt 70 ] && [ "$init_ram" -lt 95 ];
    then
        Scale_Resource 20 1 $candidate_vm 2
        sleep 3
        Calc_RAM_usage
        if [ "$init_ram" -gt 70 ] && [ "$init_ram" -lt 95 ];
            then
                Scale_Resource 20 1 $candidate_vm 2
                sleep 3
                Calc_RAM_usage
            fi
        if [ "$init_ram" -gt 70 ] && [ "$init_ram" -lt 95 ];
            then
                MigrateToServer
            fi
    fi
Calc_RAM_usage
if [ ! -z "$init_ram" ] && [ "$init_ram" -gt 95 ];
then
    Scale_Resource 40 1 $candidate_vm 2
    Calc_RAM_usage
    if [ ! -z "$init_ram" ] && [ "$init_ram" -gt 95 ];
        then
            MigrateToServer
        fi
    fi
Calc_RAM_usage
if [ ! -z "$init_ram" ] && [ "$init_ram" -lt 10 ]
then
    Scale_Resource 20 0 $candidate_vm 2
    sleep 3

```

```

Calc_RAM_usage
if [ ! -z "$init_ram" ] && [ "$init_ram" -lt 10 ]
then
    Scale_Resource 20 0 $candidate_vm 2
    sleep 3
    Calc_RAM_usage
    if [ ! -z "$init_ram" ] && [ "$init_ram" -lt 10 ]
    then
        MigrateToServer
    fi
fi
fi
i=$((i + 1))
echo "$init_cpu" >> /root/xen_logs/xenserver.log
done < /root/xen_logs/vm_name.txt
fi
rm -f /root/xen_logs/vm_name.txt
num_VM=$(xe vm-list resident-on=$host_UUID power-state=running|grep name|wc -l|cut -c 1-2)
num_VM=$((num_VM - 1))
if [ $num_VM -lt 1 ]
then
    echo "No running VM found on this server"
    echo "Server is ready to accept VMs"
else
    echo "Number of VM running on $host_node is " $num_VM
fi
running_VM=$(xe vm-list resident-on=$host_UUID power-state=running|grep vm|head -n $num_VM|cut
-c 24-100)
echo "$running_VM" >> /root/xen_logs/vm_name.txt
done

```