ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

An Efficient Live Virtual Machine Migration Technique in Cloud Computing Environment

Frezer Guteta

February 2017

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

A THESIS SUBMITTED TO DEPARTMENT OF COMPUTING OF ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

By:

Frezer Guteta

February 2017

Adama, Ethiopia

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other work and that all source of materials used for the thesis has been duly acknowledged.

Declared by:

Name: Frezer Guteta

Signature: _____

Date: _____

This thesis has been submitted for examination with my approval as university advisor

Confirmed by advisor:

Name: Frezewd Lema (PhD)

Signature: _____

Date: _____

Place and date of submission: Adama, February 2017

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

An Efficient Live Virtual Machine Migration Technique in cloud Computing Environment

By:

Frezer Guteta

Name and Signature of Members of the Examination Board

Name	Title	Signature	Date
_____	Advisor	_____	_____
_____	Chairperson	_____	_____
_____	External Examiner	_____	_____
_____	Internal Examiner	_____	_____
_____	Department Head	_____	_____
_____	School Dean	_____	_____

ACKNOWLEDGMENT

There are so many people that have contributed to this work. I owe my gratitude to all those people who have made this study possible.

First, I would like to express my sincere gratitude and appreciation to my advisor, ***Dr. Frezewd Lema*** for his patience, motivation, interest, immense knowledge in the subject area and the deft ways in which his lovingly challenge and support throughout the whole of this work knowing when to push and to let up. His guidance helped me in all the time of research and writing of this thesis. It was a pleasure working with him.

I am hugely indebted to ***Dr. Mesfin Haile*** for finding out time to reply to my e-mails, for being ever so kind to show interest in my research and for giving his precious and kind advice regarding the topic of my research.

Loving thanks to my girlfriend ***Liliy*** and my friends who played such important roles along the journey, as we mutually engaged in making sense of the various challenges we faced and in providing encouragement to each other at those times when it seemed impossible to continue.

Finally, I would like to express my deepest gratitude to my family especially to my mother's ***Teacher Frehiwet Berihun*** and my grandmother ***Ms. Atenesh Gebiresilase***; you are my model in my whole life. You have been a source of encouragement and inspiration to me throughout my life. Throughout my life, you have actively supported me in my determination to find and realize my potential.

Most of all, thanks to ***God***, the Divine, who continues to make the impossible to possible.

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF TABLE

ACRONYMS AND ABBREVIATIONS

Abbreviations	Definition
AEF	Automated Emulation Framework
ASP	Application Service Provider
BPM	Business Process Management
BRITE	Boston University Representative Internet Topology gEnerator
CBC	Characteristics Based Compression
CIFS	Common Internet File System
CPU	Central Processing Unit
EC2	Amazon Elastic Compute Cloud
E.g.	Example
ESX server	Elastic Sky X
GB	GigaByte
G.C	Gregorian Calendar
GHz	Gigahertz
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IBM	International Business Machines
IDE	Integrated Development Environment
I/O	Input Output
IP	Internet Protocol
ISCSI	Internet Small Computer System Interface
IT	Information Technology
JVM	Java Virtual Machine

KB	KiloByte
KVM	Kernel Virtual Machine
LAN	Local Area Network
MB	MegaByte
MECOM	Memory Compression
MIPS	Million Instruction Per Second
MHz	Megahertz
MPI	Message Passing Interface
MS	Millisecond
VM	Virtual Machin
NFS	Network File System
NIC	Network Interface Card
NIST	National Institute of Science and Technology
No.	Number
NS2	Network Simulation
OS	Operating System
PaaS	Platform as a Service
PC	Personal Computer
PCB	Process Control Block
PDA	Personal Digital Assistant
RAM	Random Access Memory
RQ	Research Question
QOS	Quality of Service
SaaS	Software as a Service

S	Second
SLA	Service Level Agreement
SMB	Server Message Block
SQL	Structural Query Language
TCP	Transmission Control Protocol
TTM	Time to Market
US	United State
VDE	Virtual Distributed Ethernet
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Monitor
WAN	Wide Area Network

ABSTRACT

Due to the growing number of computer nodes deployed in cloud enterprise, automatic management of electronic applications is certain. To provide various services, there will be rises in procurement, maintenance, and electricity prices. Virtualization technology is getting common in cloud computing environment, which enables the efficient use of computing resources to decreases the operating cost. Virtualization technology allows to share the same physical resources among different users and helps to reach optimum utilization of the physical resources. Hardware virtualization is the process of partitioning the physical machines into the logical machine using virtualization software called hypervisor and each logical machine is called virtual machine. Migration of a virtual machine is simply transferring the virtual machine running on a source machine to the target machine. Live virtual machine migration is migrating a running virtual machine on the source node to the destination node without interrupting any lively network connections, even after the virtual machine is copied to the target node. It is considered live because the original virtual machine is running, while the migration is in progress. An Enormous benefit of doing the live migration is the very small downtime in the order of milliseconds and small total migration time. Live migration has been expansively used in load balancing, energy reduction and dynamic resizing to increase availability and hardware maintenance. In this paper, we designed and implemented a new enhanced and efficient live virtual machine migration algorithm that reduces the total time migration and the total pages transferred and eliminates service downtime.

Keywords: - Cloud Computing; Datacenter; Virtualization; Hypervisor; Virtual Machine; Live virtual machine Migration; Total Migration Time; Total Pages Transferred; Downtime;

1. CHAPTER ONE: INTRODUCTION

1.1. Background Study

Cloud Computing is the latest technology that comprises delivering hosted services over the internet, based on a pay-as-you-go approach. It allows for the provision of a variety of business and customer services. Consumers, especially the business organizations, can extend their existing computing provision and easily scale up Information Technology (IT) facilities by consuming services available in the Cloud. There are generally three varieties of services, namely, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). There are four types of deployment approaches Private Clouds, Public Clouds, Community Clouds and Hybrid Clouds [1].

Virtualization is the key feature behind the concept of cloud computing. It is a system that provides an abstract execution environment in terms of computer hardware on top of which a guest operating system can be run. In this model, the guest is represented by the operating system, the host by the physical computer hardware, the virtual machine by its emulation, and the virtual machine manager by the hypervisor. The hypervisor is generally a program or a combination of software and hardware that permits the abstraction of the underlying physical hardware [2].

A hypervisor creates and runs virtual machines. The actual responsibility of the hypervisor is to represent computer hardware to virtual machine monitors (VMMs). Each VMM is responsible for running and monitoring exactly one guest operating system. This means that each of the guest operating systems has their own virtual platform to work on. Virtualization of computer hardware can be done with a couple of different techniques. These are paravirtualization, full virtualization, Partial virtualization and hardware assisted virtualization [2].

Live migration of a virtual machine is simply moving the VM running on a source node to the target node. It is considered live because it migrates the VM from the source node to the target node without interrupting any active network connections, even after the VM is copied to the target node. Live migration has been widely used in load balancing, energy reduction and dynamic resizing to increase availability and hardware maintenance.

1.2. Statement of the Problem

Cloud computing means delivering services to the customer based on a pay-as-you-go approach. Service providers are always working hard to deliver quality services with a minimum infrastructure cost to the customer using the visualization technology. Virtualization is abstracting the underlying physical hardware resources (e.g. Memory, CPU and NIC etc.) with the help of hypervisors to create a VM with its own OS and applications. One of the best features of virtualization is live virtual machine migration.

Live virtual machine migration is the transferring of the running VM from the source host to the target host. It is widely used in cloud clusters and datacenters for power (or energy) consolidation, load balancing, hardware maintenance and data recovery. The performance of live VM migration is dependent on the two metrics which is the service downtime and total migration time. Service down time is the time when the services are unavailable to the user during the live migration process and total migration time is the whole time taken from the start to the end of the live migration process. The value of service downtime and total migration time can be varied depend on the size of the VM to be migrated, the speed of the network bandwidth and the geographical distance between the source and the destination node. If the value of this two metrics is high the performance of the live migration process will be low otherwise the performance will be high. To address those issues mentioned above, the different cloud providers uses different live VM migration techniques.

The most popular live migration technique is the pre-copy and the post-copy approach. In pre-copy [3], first the memory pages are transferred iteratively then after some rounds, the VM at source host will suspend and its CPU state and remaining dirty pages are transferred. The minimum service downtime in pre-copy approach is 60ms(millisecond) but if the size of memory pages is huge, slow network bandwidth and large distance between the source and the target host then the number of round uses to transfer this memory pages will be large this will increase the service downtime and the total migration time.

In post-copy [4], first the source node will suspend and its execution state will be switched to the target node then all memory pages will transfer to the destination node in one round. This approach decreases the total migration time compare to the pre-copy approach because it transfers all memory pages in one round but with slow network speed and the large distance between the source node and the target host the service downtime will be high.

Finally, the pre-copy approach has minimum service downtime and high total migration time but in the post-copy has high service downtime and minimum total migration time. To deliver a QoS to the customer cloud providers must use the best live migration algorithm with minimum service downtime and total migration time. In this research, we worked to fill this gap by introducing a new live migration approach.

1.3. Research Question

The following basic research questions are used to guide the study, and they are answered in this study:

RQ1. How to optimize currently available live VM migration approaches?

RQ2. Is it possible to minimize total migration time during live VM migration?

RQ3. How to reduce the number of pages transferred during migration process?

RQ4. Is it possible to eliminate service downtime during live VM migration?

1.4. Objectives

1.4.1. General Objectives

To design and implement a new live VM migration approach with zero services downtime and minimum total migration time for cloud computing environment.

1.4.2. Specific Objectives

- To review and study related works done in live VM migration so far.
- To create cloud computing environment in CloudSim.
- To design and implement the proposed algorithm.
- To implement live VM migration in CloudSim cloud environment.
- To test and analysis the performance of the proposed algorithm.
- To evaluate the performance of the proposed algorithm comparing with some existing algorithm.

1.5. Scope and Limitation

1.5.1. Scope

This study focuses on designing and implementing a new live migration approach to migrating a running VM between the source host and target host in cloud computing environment datacenter within local area network.

This research will not address the following points:

- ✓ Comparison of the proposed algorithm with the hybrid and Post-Copy algorithm
- ✓ The distance between source and target machine is not considered to measure the performance of the algorithms.
- ✓ Migrating multiple virtual machines simultaneously.
- ✓ Migrating VM across wide area network.
- ✓ Migrating a VM in the locally blocked device (or without sharing storage).
- ✓ Migrating a virtual machine that runs a real-time application.

1.5.2. Limitation

The limitation of the study is, the experiments and implementation of the proposed algorithm are performed only in cloud computing simulation tool.

1.6. Significance of the Study

This research is significant to all people who uses cloud computing especially to the service providers, clients and academically.

1.6.1. Service Provider

Cloud service provider uses different migration techniques to migrate a running VM to a different host in their datacenter to balance the load, consolidate the energy and maintain the hardware to deliver a quality of services to their customer. Currently, available migration algorithms have at least 60ms and more service downtime and higher total migration time. This will affect the provider's business. This study will enhance their service delivery by minimizing the total migration time and service downtime. It is significant to cloud providers since it reduces the infrastructure cost, increase the performance and deliver quality services to the customer. This will make the service provider them number one choices by the clients.

1.6.2. Cloud Customers

Cloud customer uses different services delivered by the cloud providers based on their needs and they pay for the services they used only. While cloud customers use some service, simultaneously cloud provider can perform hardware maintenance to increase the quality of the service they deliver without interfering the services used by the customers. This study is significant to the clients as it helps the cloud customer to get a quality of services without any service interruption.

1.6.3. Academically

It also significant academically because it can be used as a starting point for further study.

1.7. Methodology

In this study, we are used the following method to design and implement the proposed algorithm.

- ✓ First, we identified the problem to be solved. To do this, different articles, journals, gridlines, internet sites and books are reviewed.
- ✓ Second, we identified the required parameters and design the proposed algorithm.
- ✓ Third, we prepared the experiment environment using simulation tools to conduct the experiments.
- ✓ Fourth, we implemented the proposed algorithm using java programming language in the experiment environment.
- ✓ Fifth, we performed an experiment to observe the effect of our algorithm.

Finally, we analyzed the results that we got from the experiment. We also compared and evaluated the performance of our algorithm with Pre-Copy approach.

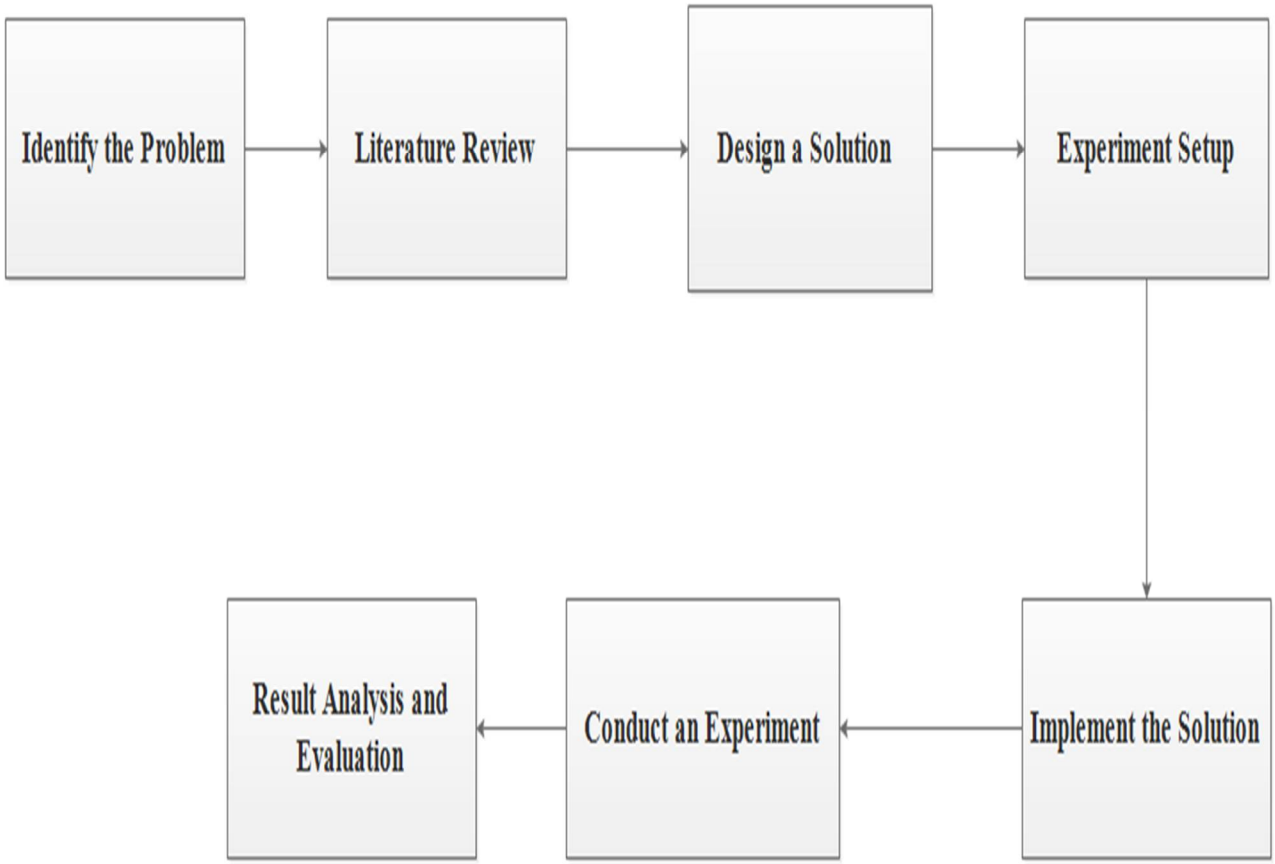


Figure 1-1: - Research Methodology

1.8. Organizations of Thesis

The rest of this paper is organized as follows:

- ✓ Chapter two describes the literature review and related work. In this part, we tried to show a pictorial view of different live VM migration methods in a cloud datacenter. We also tried to explain and demonstrate much of their features, enumerate their strengths and weakness while migrating a VM.

- ✓ In chapter three, we discuss the proposed solution. In this part, we design our algorithm to fill the gap identified in chapter one. How the solution works also explain in detail and the reason why we have chosen the method we implemented is also answered.

- ✓ In chapter four, experimental setup and implementation are described. In this part, we try to see in detail how to set our experiment and what kind of hardware and software are used to implement our proposed solution. In addition, how to implement the proposed algorithm in a cloud environment are discussed in detail.

- ✓ In chapter five, result analysis and evaluation are described. In this part, we analyzed the result we got from the experiment we performed in detail and evaluated the result by comparing with different existing approaches.

- ✓ Chapter six describes the conclusion and future work of the study. In this part, we try to conclude our work and set some future work for the future study.

2. CHAPTER TWO: LITERATURE REVIEW

2.1. Conceptual Review

2.1.1. Cloud Computing

The idea of cloud computing is not new as it goes back 50 years' from today. Professor John McCarthy, a well-known computer scientist had been thinking about time sharing since 1955 and later 1957 G.C he writes a memo for IBM 704 and in 1960 G.C for IBM 7090 computers [5]. McCarthy expected that some corporations would be able to offer computing resources to the customer through the utility business model. Later different organization starts to pay for their use of computer resource (storage, processing, bulk printing, and software packages etc.). Then after some decade's different technology is invented (Web Hosting, Application service Provider(ASP), Volunteer Computing, Online File Sharing and Social Networks) and use similar computing model.

In late 2007 G.C cloud computing appears as a new technology, which outsources different pooled, easily accessible, elastic and on-demand computing resources to the customer based on the pay-per-use model. Cloud providers offer a flexible and scalable dynamic IT infrastructure (Servers, CPU, Memory and Network etc.), QoS assured configurable computing environment and easily configurable and usable software services to the customer. Customers also can use the different services provided by the cloud providers based on their need and their ability to pay for each service they used.

There are many different definitions of cloud computing given by different scholars and institution. According to [6], the cloud is not simply the latest fashionable term for the Internet. Though the Internet is a necessary foundation for the cloud, the cloud is something more than the Internet. The cloud is where you go to use technology when you need it, as long as you need it, and not a minute more. You do not install anything on your desktop, and you do not pay for the technology when you are not using it.

In [7], Cloud computing takes the technology, services, and applications that are similar to those on the internet and turns them into a self-service utility. In the other hand, in [8], Cloud computing can be defined as a new style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet.

There is another definition of cloud computing, in [9], A computing Cloud is a set of network-enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which could be accessed in a simple and pervasive way.

The best definition of cloud computing is given by the US National Institute of Science and Technology (NIST). According to NIST [1], cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. NIST also, defines some basic cloud computing characteristics, services delivery model and deployment models as its shown in Figure 2-1.

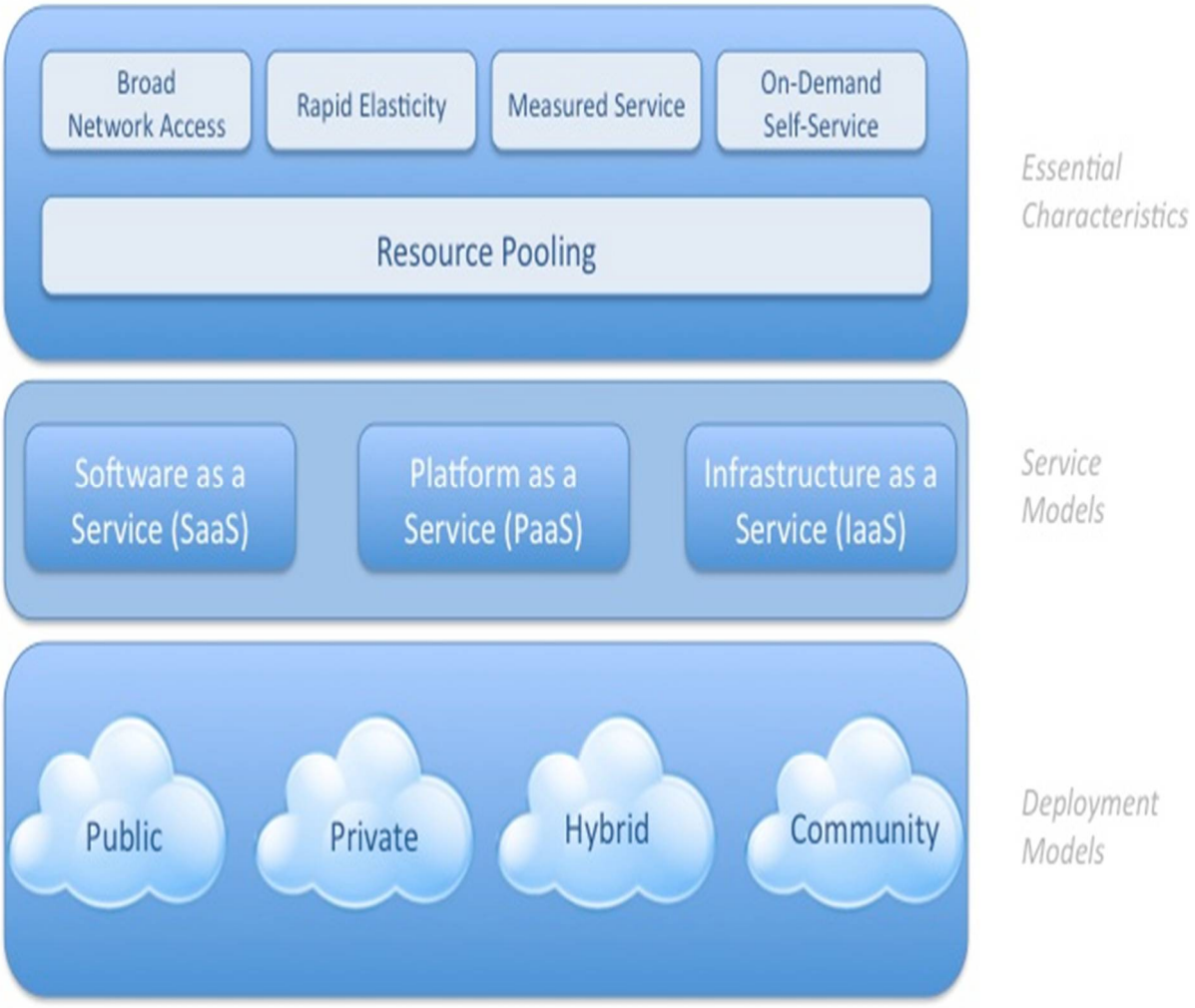


Figure 2-1: - Component of Cloud Computation [10]

2.1.2. Cloud Computing Characteristic

2.1.2.1. On-Demand Self-Service

Cloud computing provides a pool of computing resource and services to the customer on demand. Consumers can provision those resource and services automatically through an Autonomous system without any interaction with each cloud providers. Those systems enable the customers to easily customize, manage and configure their computing environment [1].

2.1.2.2. Broad Network Access

Cloud computing hosts computing resource and services beyond the client boundaries and delivers via the internet. Access to the resource is available over the network and perform through the standard mechanism. This includes different operating systems and thin and thick client platforms such as laptop, workstations, mobile phone, and PDA. As long as there is a network coverage in the area customers can access and use any resource and service they want based on their capacity to pay for each service they used [1].

2.1.2.3. Resource Pooling

Cloud computing can serve multiple users using the multitenant models. Cloud provider are pooled different resource (e.g. Storage, Memory, Processer and Network Bandwidth) to be used by multiple customers. Resources can be provisioned and released base on the customer's demand. The customers use those resources without any specific knowledge of the exact location of the resources but, they may have the high-level knowledge of the location (e.g. Country, State, and Datacenter) [1].

2.1.2.4. Rapid Elasticity

Rapid Elasticity is one of the most important features of cloud computing which enables users to rapidly and elastically accessed resource based on their demand. Elastic resources reduce the cost and decrease the time to market(TTM). Clients are not fixed to limited resources because of their ability to scale up and scale down resource based on their needs. Cloud computing resource should look like unlimited to the clients. Customers can buy any resources at any time and in any quantity with minimum cost [1].

2.1.2.5. Measured Service

Cloud customers are only charged for those services they used. Each type of services (e.g., storage, processing, bandwidth, and active user accounts) has some known metric to measure their corresponding cost. Cloud services are measured, audited and reported to both cloud customer and provider to show the transparency [1].

2.1.3. Cloud Computing Services Delivery Model

Cloud computing is about delivering all IT infrastructure as a service to the customers based on their demand over the internet. Customers can get those different services at any time without limit to their needs as long as they pay for each service they used. Cloud computing can have different service but, based on NIST [1] it only categorized into three big service categories.

2.1.3.1. Software as a Service(SaaS)

In SaaS customers, can use applications running on cloud infrastructure by subscribing into those service vendors. Once they are subscribing, they can access the application from different client devices (e.g. laptop, workstations and mobile phone etc.) via either a thin client (usually, web browsers) or programming interfaces. Clients can access the application, manage its data and user interaction and make some limited user-specific application configuration setting but, they don't control and manage the underlying infrastructure such as the network, server, operating system, storages and individual application capabilities. Cloud providers are the one who is responsible for installing, configure, maintain and manage the overall cloud infrastructure including the network, server, operation system, storage, virtual machine and the application compatibility etc. Examples of SaaS providers include Google Apps, Oracle on Demand, SalesForce.com and SQL Azure.

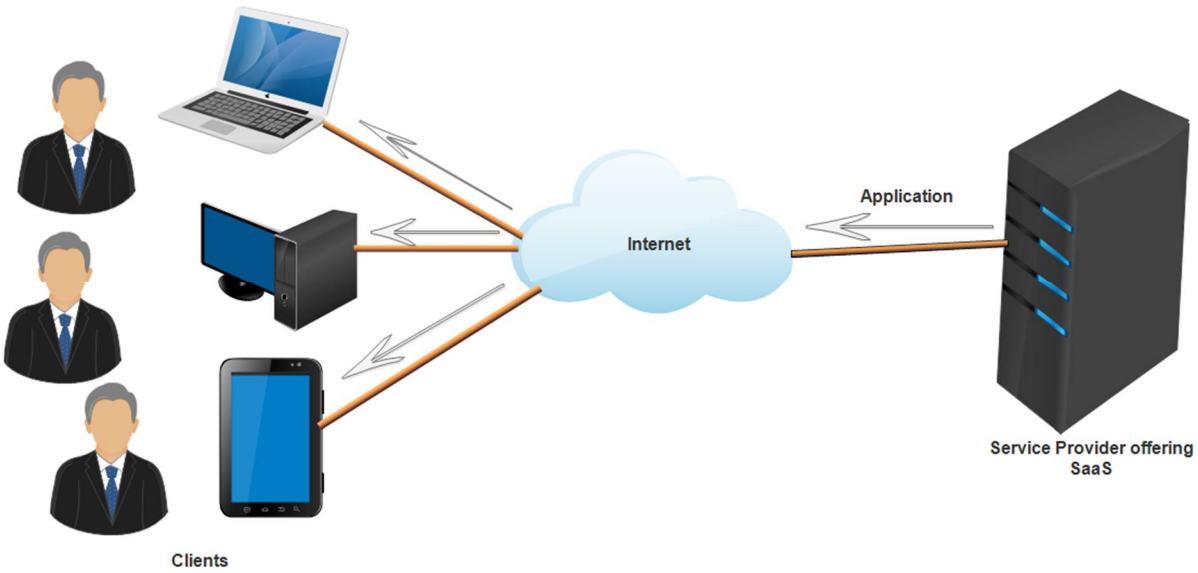


Figure 2-2: - SaaS Provides an Application or Software from Service Provider

2.1.3.2. Platform as a Service(PaaS)

PaaS is one of the most abstracted cloud service delivery models which is placed between SaaS and IaaS in the cloud stack. PaaS layer is built on top of IaaS layer. Cloud providers provide development environments such as languages, libraries, middleware, database servers, application servers, portal servers and development runtime environment etc., standards for developments and channels for distribution and payment. This enables the customer to quickly build, test and distribute their developed software product to the market or for their own uses. Customers have slight control over deployed applications and how the environment setup and configured but the underlying infrastructure (e.g. network, servers, operating systems, or storage) is taken care of by the cloud service provider. Examples of PaaS providers include Force.com, GoGrid CloudCenter, IBM Bluemix, IBM Cloudant, Amazon Relational Database Service, Google AppEngine and Microsoft Azure.

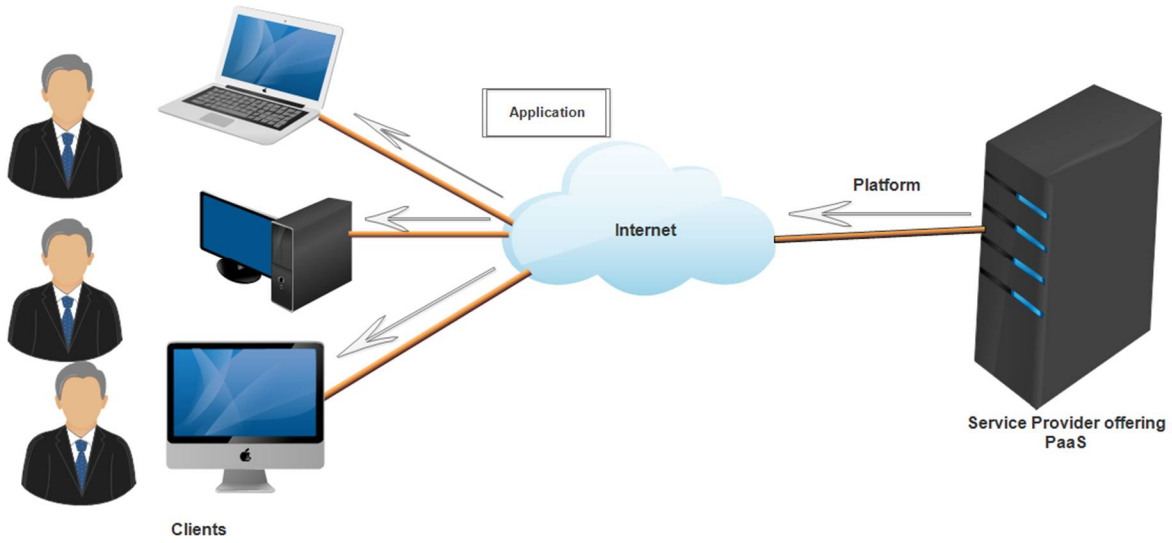


Figure 2-3: - Service Provider Provides PaaS to Allow Client to Access Computing Platform and Run Their Own Application

2.1.3.3. Infrastructure as a Service(IaaS)

IaaS is the simplest cloud service delivery model for the cloud vendors to provide their service to the customers. Cloud service provider provision computer resource like processing, storage, and networking to the client. The vendor manages and controls the computer resources including the datacenter, network equipment, computer hardware, virtualization, and automation layer. They also responsible for monitoring power, cooling, security, inventory tracking, housing, running and maintaining those

infrastructures. The customer uses the on-demand resources and installs operating system and applications based on their own choose. Consumers have control over the operating system, applications and deployed data on it and limited control over some network and security, but they don't own and manage the underlying infrastructure. Examples of IaaS providers include BM SoftLayer, IBM Cloud Managed Services, IBM Cloud Managed Backup, Amazon Elastic Compute Cloud (EC2) and Eucalyptus, GoGrid and Rackspace.

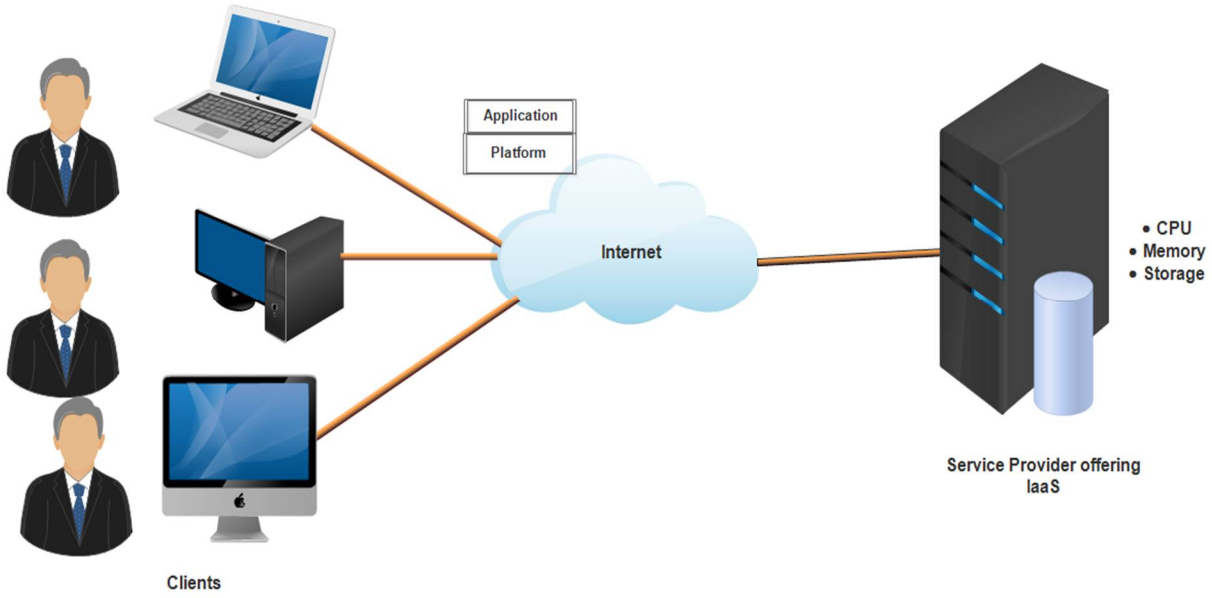


Figure 2-4: - Clients Use Their Own Platform and Applications on Top of IaaS Provided by Service Provider

2.1.4. Cloud Computing Deployment Model

Cloud can be hosted using different deployment techniques, depend on the purpose of the cloud, geographical location of the cloud and owner of the cloud to deliver a quality of service to the target end user. Based on NIST [1] definition there are four cloud deployment models.

2.1.4.1. Private Cloud

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises [1].

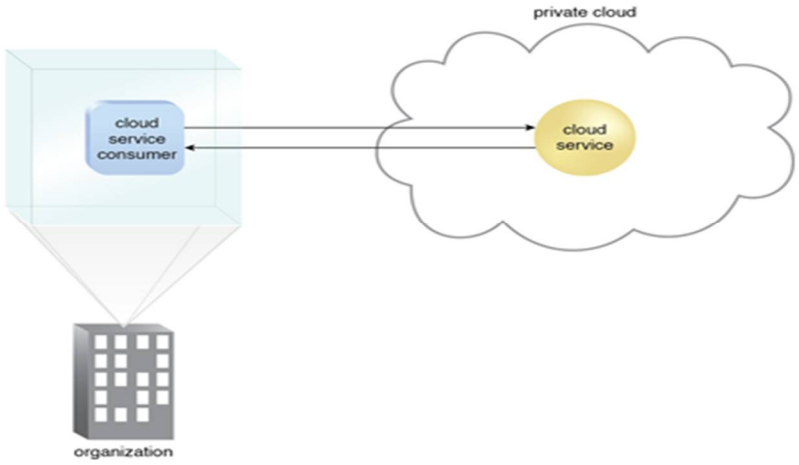


Figure 2-5: - An Organization use Private Cloud [11]

2.1.4.2. Community Cloud

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises [1].

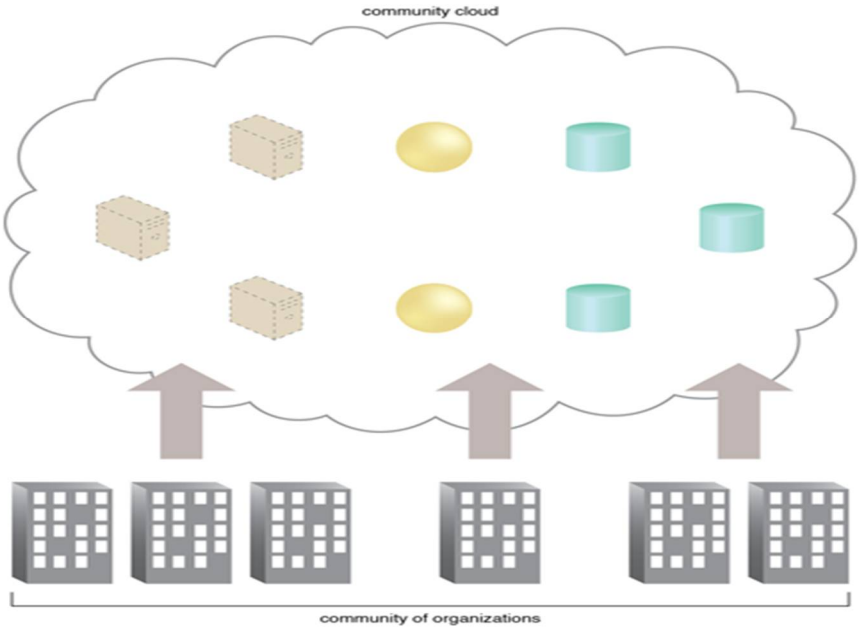


Figure 2-6: - Community Organization Uses Community Cloud [12]

2.1.4.3. Public Cloud

The cloud infrastructure is provisioned for open use by the public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider [1].

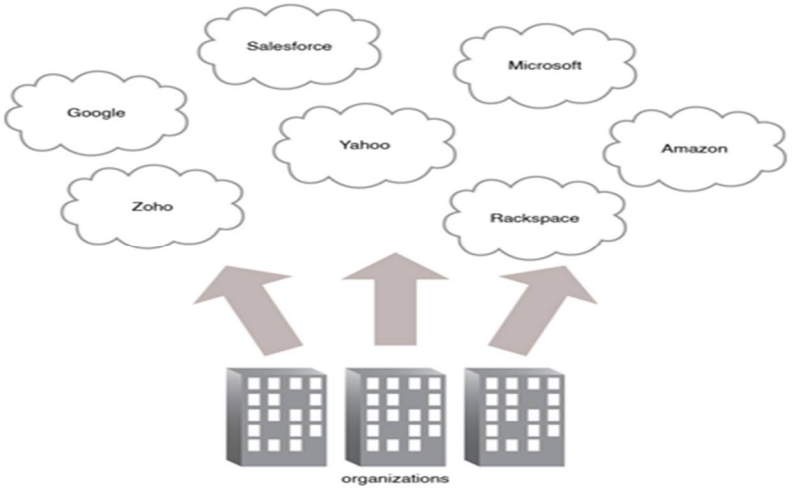


Figure 2-7: - Organizations Uses Different Public Cloud [13]

2.1.4.4. Hybrid Cloud

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds) [1].

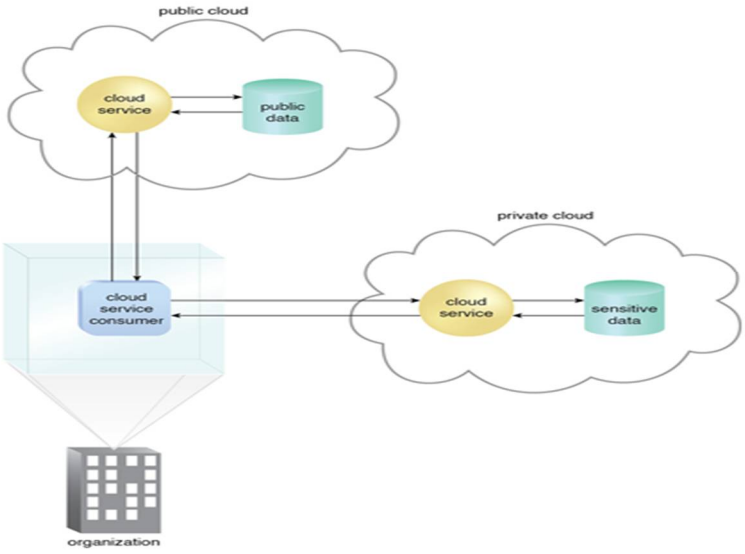


Figure 2-8: - An Organization Uses Hybrid Cloud [14]

2.1.5. Advantage of Cloud Computing

When business enterprise facilitates, their business using cloud computing technology they get a different advantage that can run their business efficiently. Among the most but not the list is listed as follow:

Cost Reduction: **-** Cloud services simply, can be accessed using a small hard disk, less memory, and more efficient processor. This reduces the cost you need to buy highly-powered or highly-priced computers. Furthermore, instead of purchasing expensive software applications, you can get most of what you need for free.

Unlimited Storage Capacity: **-** When you use cloud services you don't have to worry about the storage limitation because thanks to virtualization technology cloud computing offer virtually limitless storage.

Mobility: **-** One of the main advantages of working in the cloud is that it allows users the mobility necessary today's global marketing. Users do not need to take their documents with them. Instead, they can use the cloud, to access them whenever they have a computer or portable device and an Internet connection. Documents are instantly available from wherever they are.

Increase Data Reliability: **-** Cloud computing is a data-safe computing platform. When your personal computer is crashed, all your data is still out there in the cloud and it is still accessible.

Customization: **-** In cloud computing users, can customize the infrastructure, platforms and software's they bought as they like based on their own requirements and needs to achieve their business goals.

2.1.6. Disadvantage of Cloud Computing

Cloud computing has its own strong side, but there are some disadvantages regarding requiring a constant internet connection, security and confidentiality, data migration and less control. These issues are listed as follow: -

Requires a Constant Internet Connection: **-** When big business enterprises use cloud computing to run their business, they require a high-speed internet connection to access their documents and applications and to deliver their services to their customer. The Internet is the backbone of cloud computing. If an Internet connection is unavailable those enterprises cannot access anything, even their own documents and deliver their business to their clients. As a result, the enterprises can lose million dollars. A dead Internet connection means no work and in areas where Internet connections are insufficient or inherently unreliable, this could be a deal-breaker.

Security and Confidentiality: - Since technology has started to expand in the exponential ways in these days, cyber-crime has become a concerning issue. Cloud computing does pose the danger of increased security threats. While most companies have an up-to-date virus database, this does not make the files and information stored in the cloud protected to hackers.

Data Migration: - Nowadays' cloud users find themselves locked into a single provider, which may or may not suit their needs. This is because of the painstaking process of moving data from one cloud provider to another cloud provider.

Less Control: - Business organizations can deliver their business using clouds technologies'. But, by using these technologies', they may have less control over their own company because those infrastructures, platform, and software used to deliver their business is under the control of the third part or cloud providers.

2.1.7. Virtualization

Virtualization is a key technology behind the current concept of cloud computing, which allows multiple OS to run simultaneously on a single physical machine. It provides a way to separate (or abstract) the computing functions such as the processors, memory, and operating system from the actual physical hardware with the intent to allow the machine (or Host) to run multiple isolated operation systems (or guest virtual machine) on the top of the host operating system with the help of hypervisors (or virtual machine monitor). The hypervisor is the software that coordinates the low-level interaction between virtual machines and the underlying physical computer hardware.

Host Operating System (Host OS): **-** This is the operating system of the physical computer on which the hypervisor, guest operating system (VM) and associated applications were installed.

Guest Operating System (Guest OS): **-** This is the operating system that runs inside the virtual machine.

Virtual Machine (VM): **-** This is the virtualized physical computer environment capable of hosting the guest operating system and applications running on top of it.

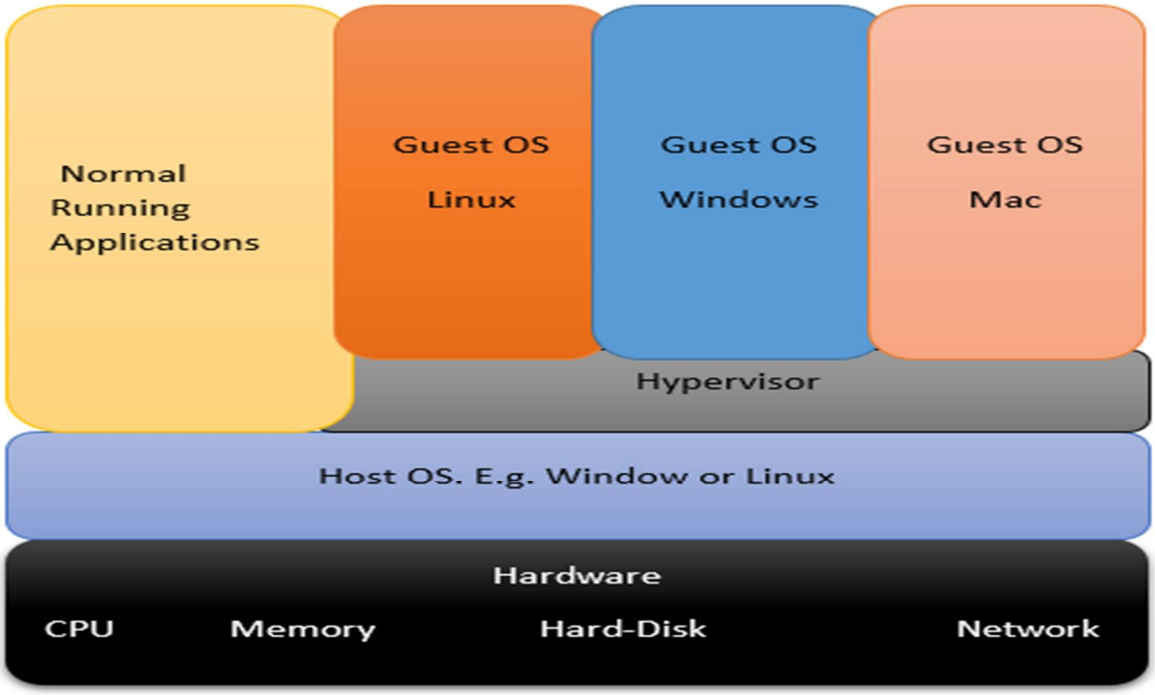


Figure 2-9: - General Structure of Virtualization

2.1.7.1. Virtualization Capabilities

Virtualization provides many capabilities to your organization. Virtualization of operating systems is used in many different computing areas [15]:

- ***Server Consolidation****:* - Many servers can be replaced by one big physical server, so hardware is consolidated, and guest operating systems are converted to the virtual machine. It provides the ability to run legacy software on new hardware.
- ***Isolation****:* - The guest operating system can be fully isolated from the Host running it. So, if the virtual machine corrupted, the Host system will not be harmed.

- ***Migration:*** - is a process to move a running virtual machine to another physical machine. Live migration is an extended feature that allows this move without disconnection of the client or the application.

- ***Disaster Recovery:*** - Virtualized guests are less dependent on the hardware, and the Host server provides snapshot features to be able to restore a known running system without any corruption.

- ***Dynamic Load Balancing:*** - is a migration feature that brings a simple way to load-balance your service across your infrastructure.

- ***Easier Software Installations:*** - Software vendors can use virtual machines to ship the entire software configurations.

2.1.7.2. Virtualization Benefits

Virtualization brings a lot of advantages while providing the same service as a hardware server. First, it reduces the cost of your infrastructure. Servers are mainly used to provide a service to a customer, and a virtualized operating system can provide the same service, with [15]:

- ***Less Hardware:*** - You can run several operating systems on one host, so all hardware maintenance will be reduced.

- ***Less Power/Cooling:*** - Less hardware means you do not need to invest more on electric power, backup power, and cooling when you need more services.

- ***Save Space:*** - Your datacenter space will be saved because you do not need more hardware servers (fewer servers than service running).

- ***Less Management:*** - Using a VM Guest simplifies the administration of your infrastructure.

- ***Agility and Productivity:*** - Virtualization provides migration capabilities, live migration, and snapshots. These features reduce downtime and bring an easy way to move your service from one place to another without any service interruption.

2.1.7.3. Virtual Machine Monitor(VMM) or Hypervisor

A hypervisor (or virtual machine monitor) is a virtualization platform that allows to create and run multiple operating systems on a host computer simultaneously. It is a layer of software that runs directly on server hardware. It controls platform resources, sharing them among multiple VM guests and their operating systems by presenting virtualized hardware interfaces to each VM guest [15]. Currently, there are two types of hypervisor which are: -

- ***Type 1 Hypervisor (or Bare-Metal Hypervisor): -*** is software that runs directly on a given hardware platform (as an operating system control program). A "guest" operating system thus runs at the second level above the hardware. Examples are Xen, VMware's ESX Server.

- ***Type 2 Hypervisor (or Hosted Hypervisor)****: -* is software that runs within an operating system environment. A "guest" operating system thus runs at the third level above the hardware. Examples include VMware server and Microsoft Virtual Server.

2.1.7.4. Virtualization Techniques

Virtualization of computer hardware can be done with a couple of different techniques, but in this paper, we try to discuss the following four techniques [15].

- ***Hardware-assisted Virtualization****: -* This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation.

- ***Full Virtualization****: -* This refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware. To make this possible, virtual machine manager is required to provide a complete emulation of the entire underlying hardware. The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform. Whereas it is the desired goal for many virtualization solutions, full virtualization poses important concerns related to performance and technical implementation.

- ***Paravirtualization:*** - This is a not-transparent virtualization solution that allows implementing thin virtual machine managers. Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified. The aim of paravirtualization is to provide the capability to demand the execution of performance critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution. This allows a simpler implementation of virtual machine managers that have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host.

- ***Partial Virtualization:*** -It provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation. Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported, as happens with full virtualization.

2.1.7.5. Virtual Machine Migration

Virtual machine migration is a key feature of virtualization technology. Migration of a virtual machine is simply moving (or transferring) the virtual machine from one physical machine (source node) to another physical machine (target node). Basically, there is two type of VM migration [3].

1. ***Offline Virtual Machine Migration:*** - The VM at source is suspended and transferred to the target host and resume there. During the migration process, the user is halted until the VM resumes at the target machine.

2. ***Live Virtual Machine Migration:*** - It is done as, while the VM is running on the source node, and without disrupting any active network connections, even after the VM is moved to the target node. It is considered live since the original VM is running, while the migration is in progress and the user is able to continuously execute during the migration. There are two relevant metrics for live migration.

    - ✓ ***Total Migration Time:*** - is the time since the live migration process starts until the VM is resumed in the destination host.

- ✓ ***Migration Downtime:*** **-** is the time during which the services and the applications running in the VM are unavailable.

There are two frequently used live virtual machine migration approaches which are called Pre-copy and Post-copy. In pre-copy approach pages of memory are iteratively copied from the source machine to the destination host, all without ever stopping the execution of the virtual machine being migrated [3]. In post-copy approach, each memory page is transferred only once [4], which is the main benefit over pre-copy approach. The pre-copy approach provides high reliability against system fault because it contains a copy of the memory pages on both sides, whereas in post-copy approach memory page can be found on one side only. We discuss the two approaches in detail in related work section 2.2.

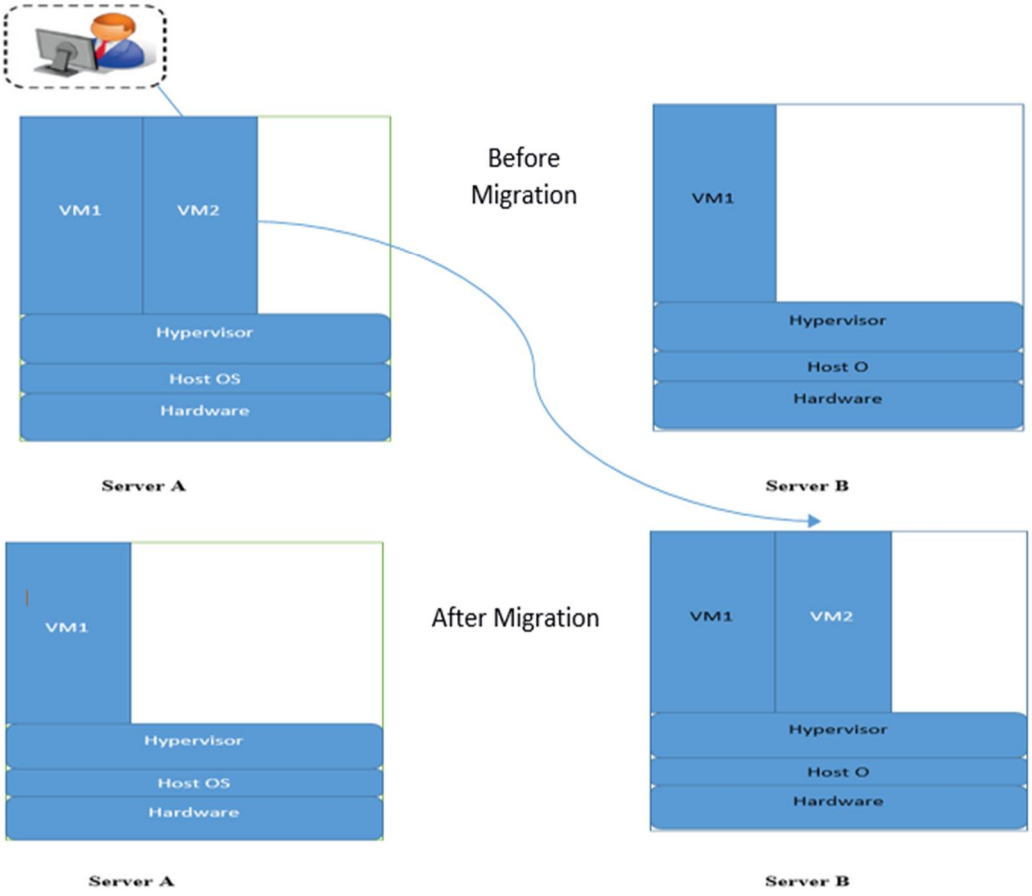


Figure 2-10: - Live Virtual Machine Migration

2.2. Related Work

Live virtual machine migration is one of the extremely powerful tools of virtualization technology. Live migration can be categorizing into two major types. The first one is non-live migration process, in this approach, the status of the virtual machine is suspended and users face service interruption during migration. On the other hand, the live migration process will keep the running states of the virtual machine and it will not lose the status of virtual machines during the migration process. Our discussion will mainly focus on live migration instead of non-live migration because it is out of the scope of our study. Generally, in live migration process the complete running states of CPU-register, memory and other devices, etc. are transferred from a source to a destination host without minimum seamless service interruption. Hence, a key issue in live migration is the total migration time and the downtime. In Data center, live migration of virtual machine performs an important role. Live migration has been widely used in load balancing, energy consolidation and fault tolerance and hardware maintenance. Nowadays, there are different live migration approach are designed and implemented by different researchers but most of them are built on the top of the two frequently used live migration approach: pre-copy and post-copy algorithm. In the following section, we try to see different kinds of literature views that enable live migration possible.

In the pre-copy approach [3], the bulk of virtual machine memory pages is copied from the source host to the destination host while the guest is still running on the source host. If some memory pages are dirtied (or modified) during this process, it will be sent in the next round. This iterative copying of memory dirty pages will continue until either a small writable working set has been identified or a predefined iteration number is reached. After the memory pages are transferred, the virtual machine at the source host will be suspended and its processor state plus any remaining dirty pages are sent to the destination host. Finally, the virtual machine will resume at the destination host and its copy at the source host will be destroyed. The time between stopping the virtual machine on the original host and resuming it on destination host is called downtime and takes a few milliseconds depend on the size of memory and applications running on the virtual machine. The goal of pre-copy based migration is to preserve downtime small by reducing the amount of virtual machine state that needs to be transferred during downtime. In pre-copy during migration process pages that are repeatedly dirtied may have to be transmitted multiple times because of this the total migration time will be increased and its effectiveness will be reduced but, it has well virtual machine downtime and application degradation. The main

advantage of this approach is the reliability and robustness because the migration process can be reverted at source host if the migration fails. Many hypervisor-based approaches such as VMware, XEN and KVM has used the pre-copy approach for live migration of virtual machines.

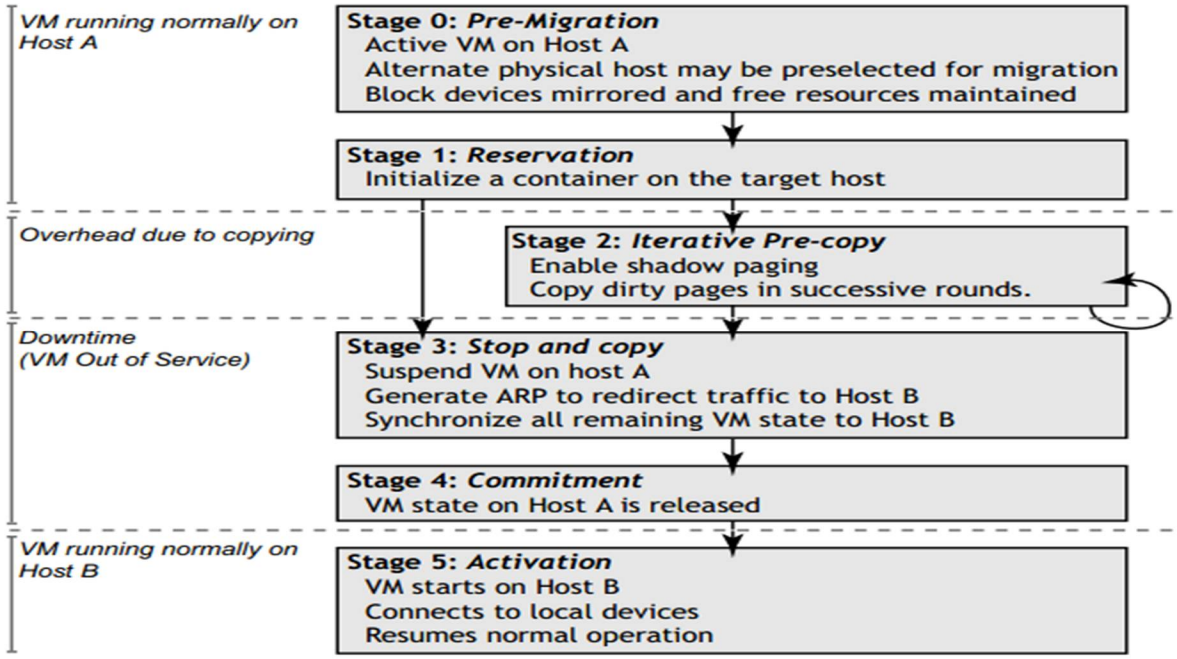


Figure 2-11: - Pre-Copy Approach Live Migration Timeline [3]

Post-copy [4] virtual machine migration, is initiated by suspending the migrating virtual machine at the source host, transmit a minimal processor state to the destination host, start the virtual machine (or guest) at the target host and then actively pushes memory pages from the source host to the destination host. Concurrently, any memory pages that are faulted on by the virtual machine at the destination host and not yet pushed are demand-paged over the network. It sends all memory page exactly once over the network during the migration process. In this process of live migration, the virtual machine at destination will immediately start responding and execution, but it might suffer from performance penalties because of network page faults. The total migration time will be based on the workloads and network link speed between two hosts, but the downtime is much higher than that of pre-copy due to the latency of fetching pages from the source node before virtual machine can be resumed on the target.

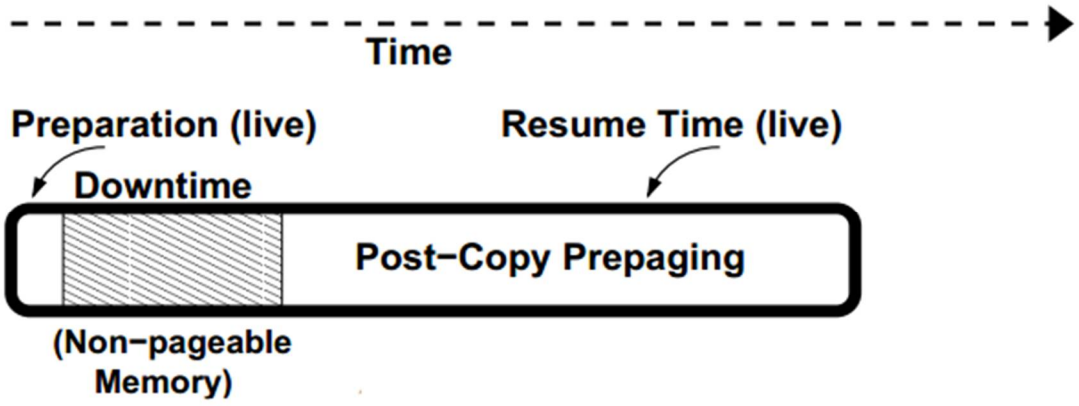


Figure 2-12: - Timeline of Post-Copy Live Migration Approach [4]

In [16] the author proposes an optimized live migration of virtual machine using CPU scheduling for the iterative pre-copy algorithm, which could limit the dirty rate of the virtual machine, thus help to improve the performance of live migration process. The main idea of the algorithm is when a VM writes memory too fast to perform the pre-copy. It schedules the CPU time for this VM to a suitable percentage so that the dirty rate could be adjusted to a small enough value. The downtime and the total time migration process could be decreased to acceptable ranges with some acceptable CPU overhead.

MECOM [17] is introduced memory compression technique into live VM migration. This schema is based on memory page characteristics, MECOM design a memory compression algorithm for live migration of VMs. The total migration time and downtime are both reduced significantly because the smaller amount of data is transferred but compression operations cause performance bottleneck of live migration

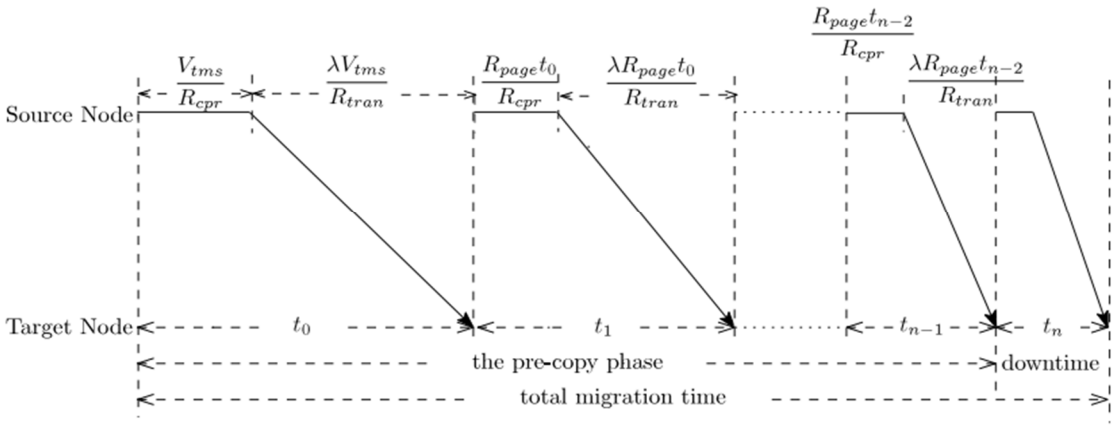


Figure 2-13: - Timeline of MECOM life migration approach [17]

In [18] the authors proposed a framework called a work set prediction algorithm based on the least recently used algorithm for pre-copy live virtual machine migration approach to reducing the amount of data transfer and iteration time during migration. The framework includes the pre-processing phases in addition to the push, stop and copy phases of the pre-copy algorithm. Work set prediction approach collects the most recent used memory pages as the working set list, the least recent used memory pages as the inactive list in pre-processing phase and it migrates the inactive list in push phase. Finally, it migrates the Working set list in stop phase. Based on this paper the framework can reduce the total migration time in the live migration of virtual machines but the prediction time is a performance bottleneck of the additional overhead introduced by prediction operation.

In another paper [19], an algorithm is proposed by modifying the pre-copy approach. Pre-copy is the widely accepted live virtual machine migration approach but the problem with this approach is that the total migration time will increase to a very high value as the page-dirtying rate increases. This approach creates a log record to store those modified pages and compress the log record using Run Length Encoding Technique. It reduces the downtime and total migration time by transferring the pages that are not recently used and sending the log records of modifications instead of resending the dirty pages.

Hybrid migration [20], this approach consists the very best future of pre-copy and post-copy live migration approach. The main goal of this approach is to balance the performance of pre-copy and post-copy algorithm. In this approach, migration is initiated at source host by transferring the most frequently accessed memory pages like in the pre-copy schema. After that, the execution state will be transferred to the target host and resume there. Finally, like the post-copy approach, the remaining memory pages are copied over the network. This approach reduces the performance degradation of the virtual machine to some extent because of the small number of memory pages transferred during the pre-copy of a hybrid approach. The number of the round in pre-copy of hybrid approach can be defined by the number of round or time for transfer memory pages. The hybrid scheme may take extra total migration time than the post-copy approach. In hybrid approach choosing the correct set of memory pages for transfer from source to the destination host is very challenging.

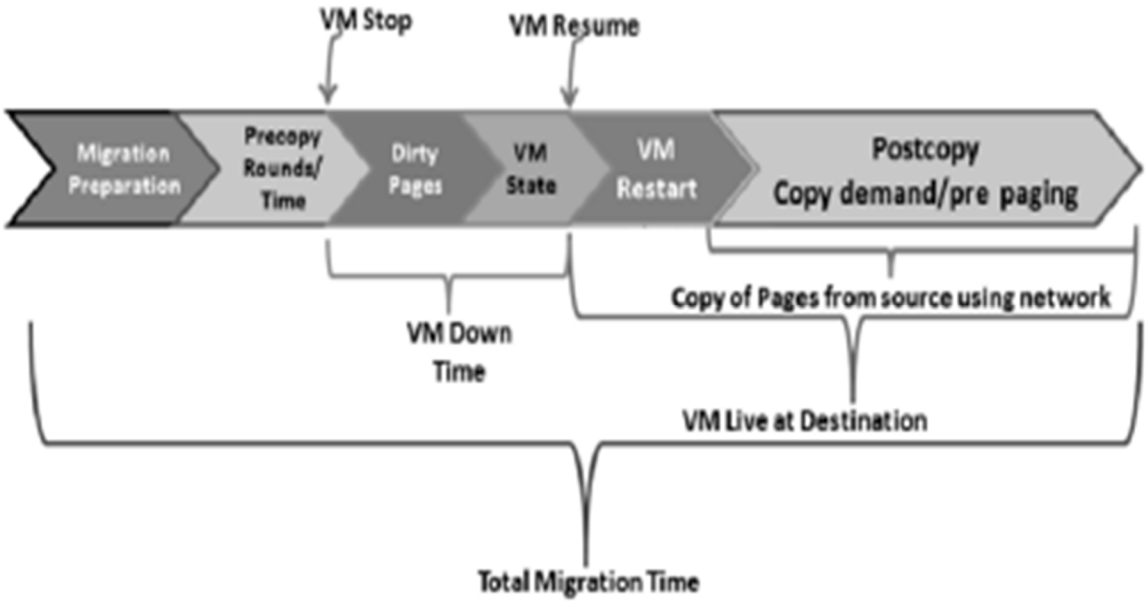


Figure 2-14: -Migration process of hybrid [20]

In [21] the author attempts to adapt the live virtual machine migration algorithm of KVM hypervisor. They improved the stop stage condition of the KVM live migration method to enforce that the algorithm enters the stop-and-copy stage in case the dirty page rate practiced by the VM at a given round is high compared with the transfer rate. Based on this paper, the adaptive version decreases the downtime produced by the live migration process compared with an original version of KVM that operates with an enlarged downtime to ensure convergence.

In another paper [22], a new live migration algorithm is proposed by modifying the existing optimized pre-copy algorithm. They add a parameter and set some threshold value and combine it with CBC (characteristic Based Compression) algorithm. In this paper data transferred in each round are first compressed by the compression algorithm and when arriving at target compressed data are decompressed. Compression time may create performance bottleneck but they removed this bottleneck by introducing multithreading technique to parallelize compression. Based on this paper both total migration time and down time will be reduced. However, they didn't implement this algorithm in any simulation tool or real environment experiment.

Table 2-1: - Related Work

Title	***Author***	***Drawback***	
		Downtime	***Total migration time***
Live Migration of Virtual Machines	Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield	Yes	Yes
Post-Copy Live Migration of Virtual Machines	Michael R. Hines, Umesh Deshpande, and Kartik Gopalan	Yes	Yes
Optimizing the live migration of virtual machine by CPU scheduling	Hai Jin, Wei Gao, Song Wu, Xuanhua Shi, Xiaoxin Wu and Fan Zhou	Yes	Yes
MECOM: Live migration of virtual machines by adaptively compressing memory pages	HaiJin, LiDeng, SongWu, XuanhuaShi, HanhuaChen and XiaodongPan	Yes	Yes
An Optimized Approach for Live VM Migration using Log Records	S, Anju Mohan, and Shine	Yes	Yes
Live Virtual Machine Migration with Efficient Working Set Prediction	Thein, Ei Phyu Zaw, and Ni Lar	Yes	Yes
A Hybrid Approach to Live Migration of Virtual	Shashank Sahni and Vasudeva Varma	Yes	Yes

Machines			
Adaptive Downtime for Live Migration of Virtual Machines	Francisco J. Clemente-Castell´o, Juan Carlos Fern´andez, Rafael Mayo, Enrique S. Quintana-Ort ´	Yes	Yes
Efficient Virtual Machine Migration in Cloud Computing	Megha R. Desai and Hiren B. Patel	Yes	Yes

From the above discussion, there are different researchers that are trying to migrate virtual machines seamlessly live from one source machine to the target machine with minimal service downtime and total migration time. Thus, researchers are attempted to reduce the total migration time and downtime by introducing different techniques.

But, as we see in the above discussion and Table 2-1 almost all currently available live migration techniques are suffered from service downtime and high total migration time. Our study intended to design and implement a new live virtual machine algorithm to remove the service downtime and minimize the total migration time effectively.

3. CHAPTER THREE: PROPOSED SOLUTION

From the above literature review, we understand that most papers are worked on live migration of virtual machine using pre-copy or post-copy approach. And all other papers are trying to optimize that two approach by introducing the concept of memory page compression, work set prediction to reduce the number of iteration in pre-copy, logo record to track dirty page and CPU scheduling. Here, we try to propose a pre-copy based new algorithm which is more optimized compare to the other stated on the related works.

3.1. Design

3.1.1. Design Overview

In this section, we discussed the logical phase we used to migrate operating system instance from source to target host using our proposed approach that is summarized in Figure 3-1.

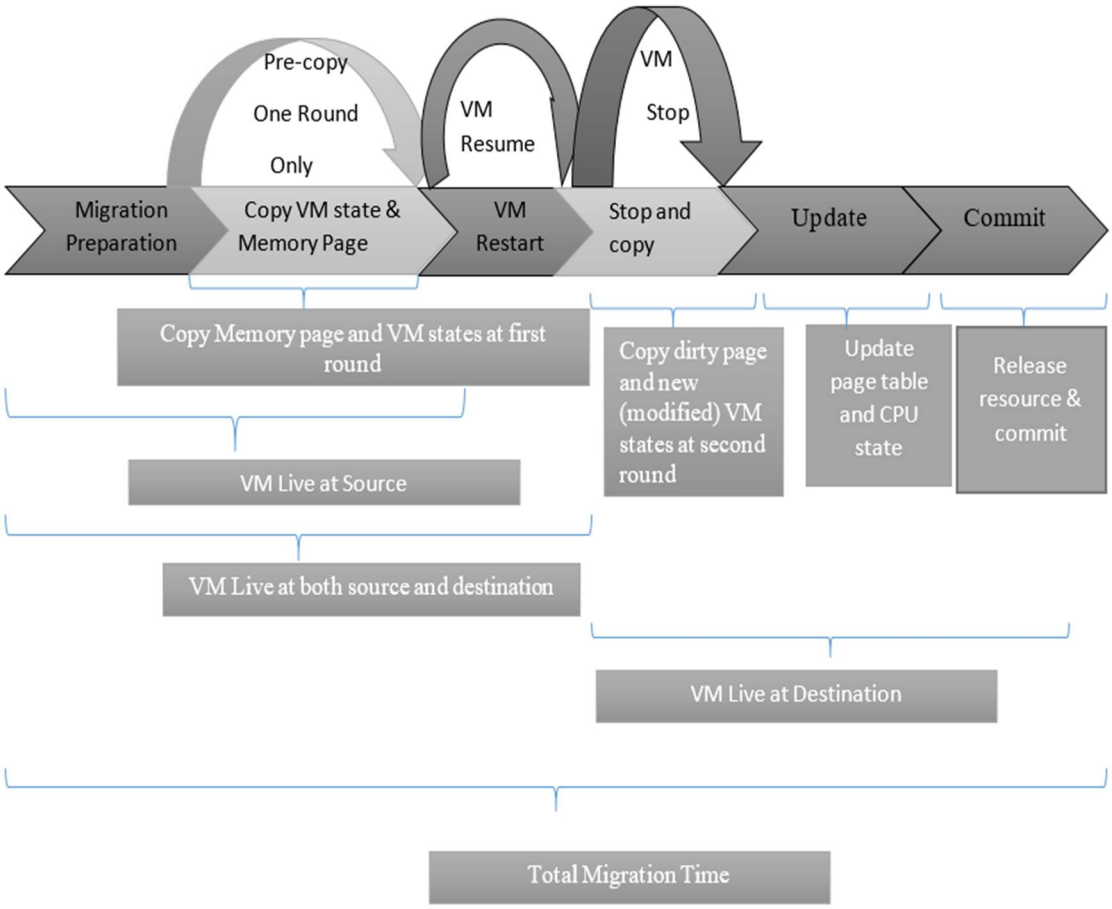


Figure 3-1: - Proposed Approach Timeline

Phase 1: Pre-Migration: - In this phase, everything that makes the migration process successful will be prepared before the migration process starts. First, a TCP/IP network connection needs to be established between the source node and the destination node because live VM migration will occur over TCP/IP connection. Second, the source and target host need to be configured to use some sort of shared storage (NFS, SMB, CIFS, ISCSI). Third, at the target host need to create a guest configuration that exactly matches the hardware settings (e.g. Memory, Processer and Network etc.) of the guest that is currently at source host. After those requirements mentioned above are meet the migration process can start with an active VM on the source machine.

Phase 2: Run and Copy: - In this phase, the VM memory pages and CPU states are parallelly copied from the source host to the destination host while the VM is executing on the source machine. The VM at the source node will continue to run until the copied memory pages and its associated CPU states are completely transferred and properly resume at the target machine. As a result, we able to eliminate the occurrence of service downtime during the migration process.

Phase 3: Resume(Activate): - Once both memory pages and CPU state are transferred completely to the target host and successfully resume there, the guest OS at the source node and the transferred guest OS at the target host will run parallelly for some time until the migration process enters the stop and copy phase. In this phase, there are two conditions needs to be checked before moving into the next phase.

1. If the migrated VM does not activate successfully at the target machine, then the VM will continue to execute on the source host. This gives the advantage of reliability.

2. If the VM does activate successfully at the destination host, then the active VM at source host will be no longer needed and the migration process will be entering the next phase which is the stop and copy.

Phase 4: Stop and Copy: - Once the VM is resumed at the target machine, the active VM at the source machine will be no longer needed. Because of this the running VM at source node will be suspended and start to copy the modified (or dirty) memory pages and new CPU states to the target host.

Phase 5: Update Page and CPU State: - At this phase, the VM at target node will continue its execution but the VM at the source node will remain suspended. The running VM at the destination host will start

to update its memory page table and synchronize CPU state with those finally copied dirty (or modified) memory pages and new CPU states.

Phase 6: Release and Commit: - This is the final phase in this live VM migration approach. In this phase, the VM at source machine is suspended and no longer needed but it uses resources (e.g. Memory, Processer, and Networks etc.) of the source host. To free those resources the VM at the source host will be destroyed. In the other hand, the destination host becomes the primary host and the copied VM will continue its execution and committed to that host.

3.2. Migrating Memory and CPU-Register-State

In the live migration of virtual machine, only memory and CPU state need to be transferred from the source host to the target one. Currently transferring memory pages and CPU states dominates the performance of live migration. Basically, there are two metrics to measure the performance of live migration process which is the total migration time and service downtime. Total migration time is the whole time from beginning to the end of the migration process. Virtual machine service downtime is the time the entire virtual machine service unavailable. In this section, we will discuss how the proposed approach can transfer memory pages and CPU state from source to target machine.

3.2.1. Memory Page Migration

In this proposed approach transferring of memory pages from the source node to target node takes place in four different phases.

1. ***Run and copy phase: -*** Memory pages are transferred from source to the destination host over the network while the VM at the source node is running. In this phase, unlike pre-copy approach which transfers memory pages iteratively memory pages are copied only once parallelly with its corresponding CPU states. This reduces the number of iteration pre-copy approach uses to transfer memory pages from the source to the target machine because of this the total migration time will be minimized. Once the VM memory pages are completely copied to the destination node with its associated CPU state the VM will be resumed there.

2. ***Stop and copy phase: -*** Once the VM start at the destination host the VM at the source host will be no longer needed because of this the virtual machine at the source will be suspended and those

modified (or dirtied) memory pages are copied. To identify those dirty pages, we adapt pre-copy approach of identifying modified memory pages. The pre-copy approach uses dirty bitmaps to identify those modified memory pages.

3. ***Update phase:*** **-** In this phase, we have two type of memory pages one is those memory pages that are copied in the first round and currently in use by the VM at the target node and on the other hand those dirty memory pages copied during stop and copy phase. Those memory pages of the active VM at destination host will be updated with those dirty memory pages while the VM is executing.

4. ***Pull phase:*** **-** The VM is running at the destination node and, those memory resource occupied by the VM at the source node will be released.

3.2.2. CPU-Register-State Migration

In this proposed approach transferring of CPU state from the source node to target node is performed in the same way as the memory migration process in four different phases.

1. ***Run and Copy Phase:*** **-** After the guest, OS memory pages are completely transferred to the target node. The process of transferring CPU state will begin immediately. This transferring of the process will be done using two threads, one for generating an interrupt and the other for copying the saved state.

   The first thread will generate an interrupt because every time an interrupt occur the system need to save the current context of the process running on the CPU [23]. This saved context will restore after processing the interrupt is done (see in Figure 3-2). A process is a program in execution. The context of the process will represent in the PCB (Process Control Block). A PCB is a repository which contains any pieces of information associated with a specific process (Content of PCB shows in Table 3-1). In general, this means the current state of the CPU will save in ready queue in the form of PCB, whether it is in kernel or user mode and restore the saved context to resume operation.

The second thread will copy the current state of the CPU to the target node. When the CPU forced to process an interrupt, it saves the context of the running process in the form of PCB into the ready queue. This thread will create a log record queue and copy the list of PCBs from the ready queue. After the CPU completes processing an interrupt it restores the interrupted process and continues its execution. Finally, the threads will send the log record queue to the target node. This VM will continue to run until the copied VM will start on the destination machine.

2. ***Stop and Copy Phase: -*** When a list of PCB is copied into the log record queue a bit is mapped alongside with them, bit 0 is for old processes PCB and bit 1 for new processes PBC. This bit uses to differentiate the old process from the new one. After the VM is started successfully at the target host the VM at source host will stop running and start to copy those new processes PCB only.

3. ***Update Phase: -*** Once all memory pages and CPU states are copied to the target node and the VM is started. The CPU will update its state with new processes PCBs copied during the stop and copy phase. To do this there are two threads, one is for generating an interrupt and the other is to update the ready queue with that new list of processes PCB.

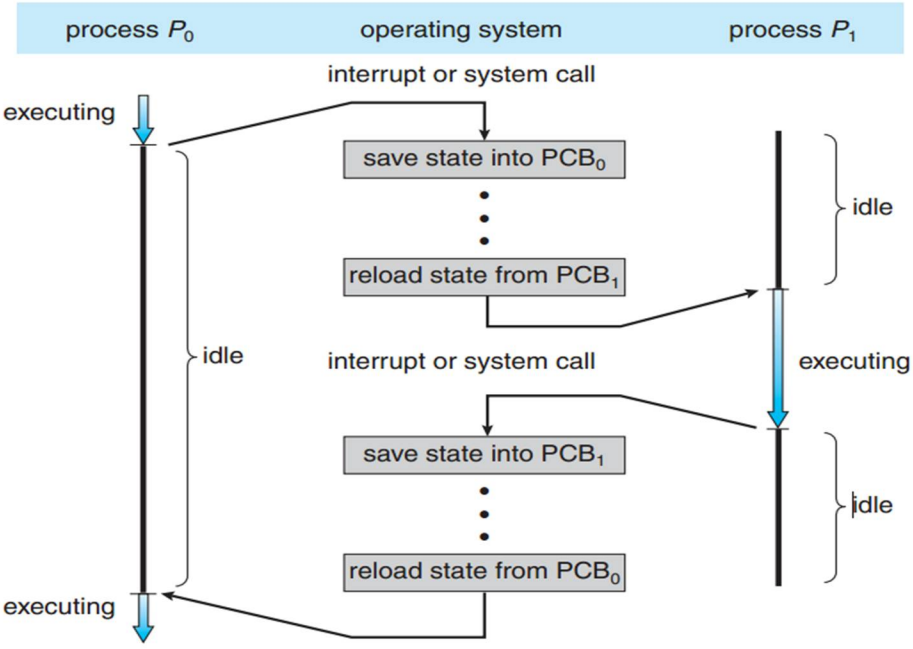


Figure 3-2: - Diagram Showing CPU Switch from Process to Process [23].

Table 3-1: -Some of the Fields of a Typical Process Table Entry [24]

Process management	***Memory management***	***File management***
Registers	Pointer to text segment info	Root directory
Program counter	Pointer to data segment info	Working directory
Program status word	Pointer to stack segment info	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameter		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

4. ***Pull Phase: -*** The VM is running at the destination node and, those CPU resource occupied by the VM at the source node will be released.

3.3. Pseudo Code and Flow Chart of Proposed Approach

In this section, we present the pseudocode and flowchart of the proposed live virtual machine migration algorithm.

Pseudo Code of Proposed Algorithm

Input:

List: list of memory pages {P1, P2, P3, P4, P5, ... Pn}

Queue: queue of processes PCB {PCB1, PCB2, PCB3, PCB4, PCB5, ... PCBn}

Output:

A successful running VM at Source Machine.

Begin:

Step 1: Pre- Migration

1.1. Establish a connection between the source and destination host.

1.2. Create Network Attached Storage for Source and Target Host.

1.3. Create a container for the target host same as the source host.

Step 2: Run and Copy

2.1 Page Transfer

2.1.1. for i=1 to n do

2.1.1.1. send(List[i]);

2.2 CPU State Transfer

2.2.1 Create LogQueue;

2.2.2 threadInterrupt ();

2.2.3 threadCopy:

2.2.3.1 For i=1to n do

2.2.3.1.1 LogQueue.push(Queue[i]);

2.2.3.2 send(LogQueue);

Step 3: Start VM at Target Machine

3.1 if (!isResume())

3.1.1 goto step 2;

3.2 else

3.2.1 StartVM ();

Step 4: Stop and Copy

4.1. Transferring Dirty Pages

4.1.1. if (List.isModified ())

4.1.1.1 Page $_{CHANGE=}$ (Page $_{ORIGINAL}$) $_{XOR}$ (Page $_{DIRTY}$);

4.1.1.2 send (Page $_{CHANGE}$);

4.2 Transferring Modified CPU States

4.2.1 if (Queue.isModified ())

4.2.1.1 PCB $_{CHANGE}$ = (PCB $_{ORIGINAL}$) XOR (PCB $_{MODIFIED}$);

4.2.1.2 LogQueue.push(PCB $_{CHANGE}$);

4.2.1.3 send (LogQueue);

Step 5: Updating Page and CPU state at Target Machine

5.1 Page $_{DIRTY}$ = (Page $_{ORIGINAL}$) XOR (Page $_{CHANGE}$);

5.2 threadInturrupt();

5.3 thteadUpdate:

5.3.1 PCB $_{MODIFIED}$ = (PBC $_{ORIGINAL}$) XOR (PCB $_{CHANGE}$);

Step 6: Release and Commit

6.1 release (Memory, CPU);

6.2 destroySourceVM (VM);

6.3 commitToTargetVm (VM);

End:

Flow Chart of Proposed Algorithm

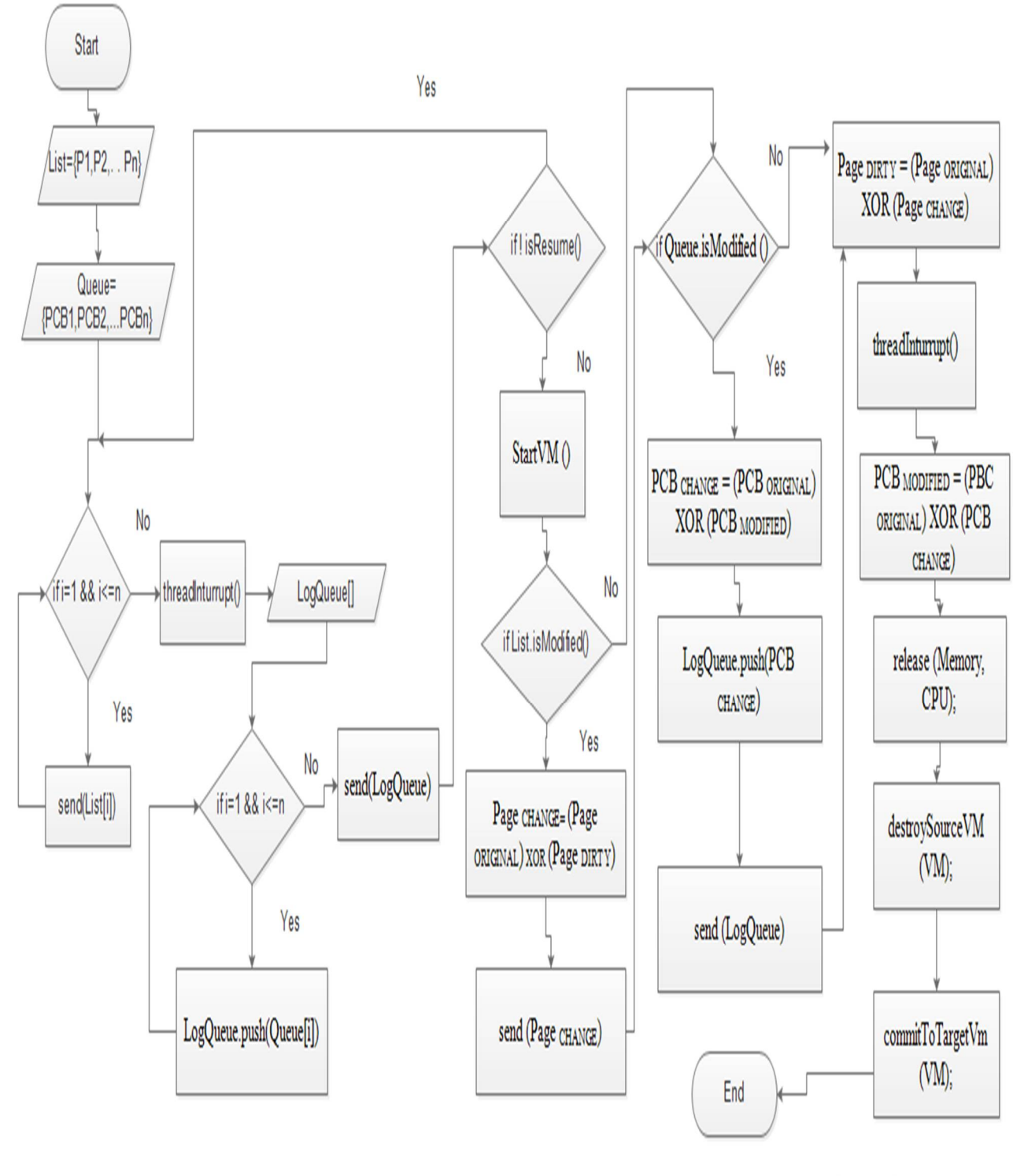


Figure 3-3: - Proposed Algorithm Flow Chart

3.4. Reliability

During live VM migration, either the source or destination node can fail. In both pre-copy [3] and proposed approach, failure of the target node does not matter because the source node still contains a whole up-to-date copy of the VM's memory and processor state and the VM can be recovered if necessary from this copy. But, failure of the source node is different in both cases.

In the pre-copy failure of source node indicates permanent loss of the VM itself but in our proposed approach this scenario is a little bit different. In the proposed approach failure of source node could permanent loss of the VM itself only if, the source node fails before or during resume phase. Otherwise, the VM could continue to run at the destination node while the source node fails because of this, the reliability of live VM migration is increased.

4. CHAPTER FOUR: EXPERIMENTAL SETUP AND IMPLEMENTATION

4.2. Experiment Setup

This section presents the experimental environment we used to conduct this research. Conducting experimental research for cloud computing using real world infrastructure is extremely time-consuming due to their sheer scale and complexity because of this, we used a cloud simulation tool. In this section, we discussed different popular cloud simulation tools and we make a comparison between that simulation tool to choose the best one. Also, we describe the hardware and software requirement we used in this study.

4.2.1. Software Requirement

The software we used to conduct this experiment is listed below.

- ✓ Operating System: Microsoft Windows 10Pro
- ✓ IDE: NetBeans IDE 8.1
- ✓ Programming Language: Java
- ✓ Simulation Tool: CloudSim 3.1.3

4.2.2. Hardware Requirement

This is the hardware requirement we used to conduct this research.

- ✓ Processer: Intel(R) Pentium(R) CPU B950 @ 2.10GHz, 2100 Mhz, 2 Core(s), 2 Logical Processor(s).
- ✓ RAM: Installed Physical Memory (RAM) 4.00 GB
- ✓ System Type: x64-based PC
- ✓ Hard Disk: 500GB

4.2.3. Simulation Tool

Cloud computing has been one of the rapidly growing parts in IT industries. Simulation-based approaches become popular in industries and academics to evaluate cloud computing systems, application behaviors, and their security. Simulation is a technique where program models and the behavior of the system can have calculated by the interaction between its different entities using mathematical formulas. Simulation

is used for providing a controllable and repeatable environment to the services. Simulation has different advantages like it reduces cost, leads to the better result, Evaluation of risk at any early stage, Ease of use and customization. Several simulators have been specifically developed for performance analysis of cloud computing environments including CloudSim [25], GreenCloud [26], NetworkCloudSim [27], CloudAnalyst [28].

4.2.3.1.Comparison of Simulation Tool

In this part, we discussed some of currently available cloud computing simulation tools and comparison between them is done.

1. ***CloudSim:*** **-** CloudSim [25] is a toolkit for the simulation of Cloud computing environments developed in the clouds Laboratory at the Computer Science and Engineering Department of the University of Melbourne, Australia. it is a generalized and extensible simulation toolkit that enables seamless modeling, simulation and experimentation of emerging cloud computing system, infrastructures and application environments for single and internetworked clouds. It aims to provide cloud computing researchers with an experimental tool to conduct cloud computing-related research. It supports the modeling and simulation of various cloud computing components, including power management, performance, data centers, computing nodes, resource provisioning, and virtual machines provisioning. CloudSim allows simulation of scenarios modeling IaaS, PaaS, and SaaS services. CloudSim is a layered design framework written in Java and was initially built on top of SimJava and GridSim. The CloudSim architecture is shown in Figure 4-7.

2. ***CloudAnalyst:*** **-** CloudAnalyst [28] is developed at the GRIDS laboratory at the University of Melbourne and designed for some specific goals. It is a graphical simulation tool which built on top of CloudSim toolkit. This simulator can be applied to examining the behavior of large scaled Internet application in a cloud environment. It separates the simulation experimentation exercise from a programming exercise. It also enables a modeler to repeatedly perform simulations and to conduct a series of simulation experiments with slight parameters variations in a quick and easy manner. This simulation tool supports the evaluation of social network tools, based on the geographic distribution of users and data centers. It also configures all parameters of simulation at a high level of performance and accuracy which show the result in the form of chart and tables which are easy to understand. The main features of CloudAnalys are easy to use, graphical user

interface, the ability to define a simulation with a high degree of reconfigurability and flexibility, repeatability of experiments, graphical output and use of consolidated technology and ease of extension. The CloudAnalyst architecture is shown in Figure 4-1.

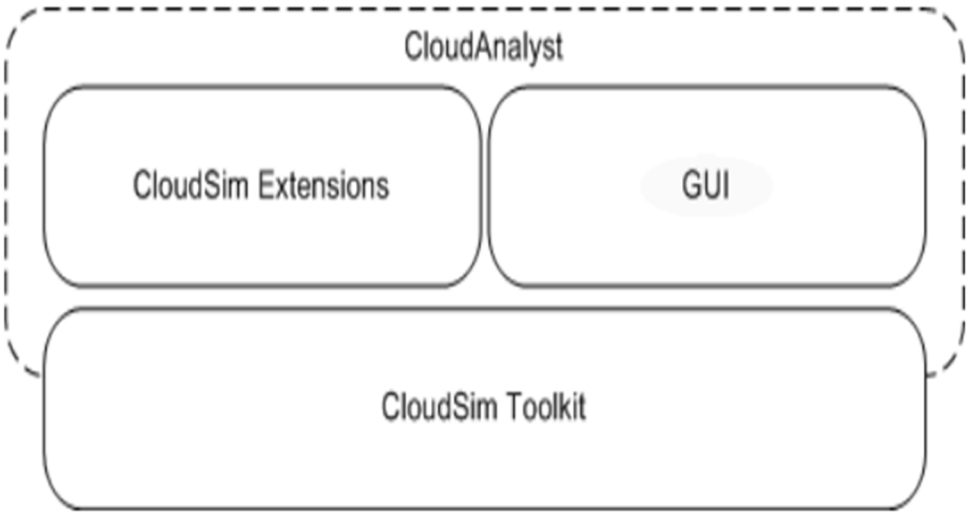


Figure 4-1: -Architecture of CloudAnalyst [28]

3. ***CloudReport:*** **-** CloudReport [29] is another tool which is built upon CloudSim framework, but it improves many features of CloudSim such as a user-friendly graphical user interface, running multiple simulations at the same time, and display simulation results in chart and table. It also provides information about VM allocation, energy consumption, and any other user-defined characteristic. It provides different features for the researcher to play the role of service providers and users. It enables the user to set the number of virtual machines each customer owns, a broker responsible for allocating these virtual machines and resource consumption algorithms, set the number of computational nodes and their resource configuration, which comprises processing capacity, the amount of RAM, available bandwidth, resource utilization and execution time. Also, it allows service providers to evaluate their cloud environment before leasing the services to the users. It doesn't require programming skill to simulate cloud computing environment. The CloudReport architecture is shown in Figure 4-2.

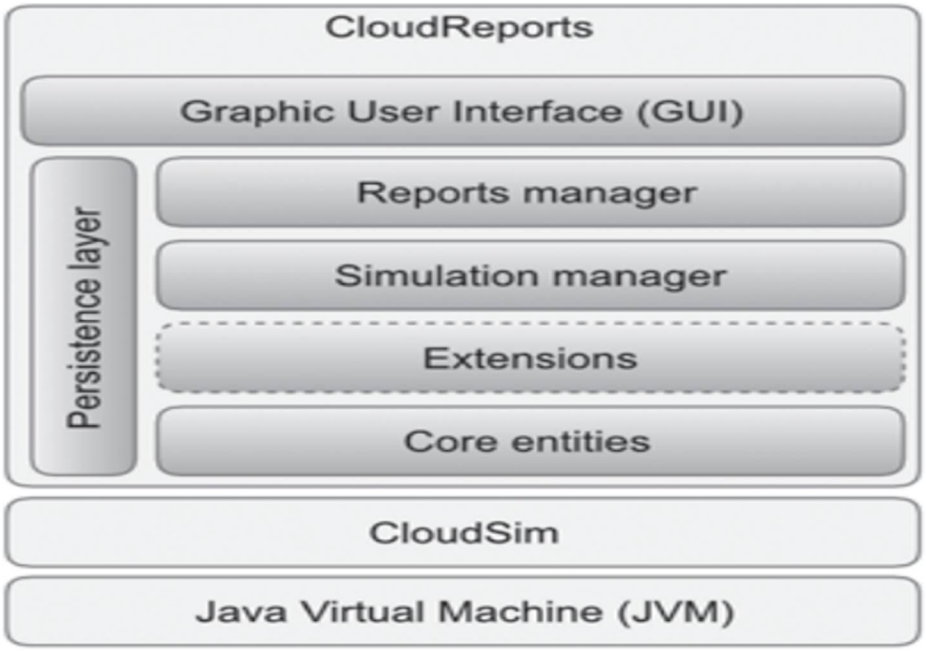


Figure 4-2: - CloudReports Organization Overview [29]

4. ***CloudExp*: -** CloudExp [30] is a modeling and simulation environment which introduced a specialized mobile cloud computing experimental framework and integrates several new features and important components to CloudSim. The added components include a more-detailed workload generator, modules for MapReduce, modules for mobile cloud computing, extensions for SLA and BPM, a wide range of network topologies, and a Graphical User Interface (GUI). A GUI extension enables users to conduct research experiments by creating the main components of a cloud system. This simulator allows researchers to study the communication cost between users and clouds. In CloudExp, cloudlets are created using a simple drag and drop. Additionally, it conducts various mobility scenarios for mobile devices. In CloudExp, the users need to configure the workplace properties and then decide the number of users, a number of cloudlet in that region, the cloudlet type (fixed or mobile), and other properties of cloudlet. When the simulation complete, CloudExp results are saved as Excel sheets to simplify the analysis of results.

5. ***GreenCloud:* -** GreenCloud [26] is an enhancement of CloudSim and considered as an extension of the network simulator Ns2. It is designed to prove the green cloud computing. The motivation behind GreenCloud development to interact and measure cloud performance is the lack of detailed simulators, and having no provisioning system for analyzing the energy efficiency of the

clouds. This approach is a sophisticated packet-level simulator for energy-aware cloud computing data centers with a focus on cloud communications. It provides a detailed fine-grained modeling of the energy consumed by the data center IT equipment such as computing servers, network switches, and communication links. It can be used to develop novel solutions in monitoring, resource allocation, workload scheduling as well as optimization of communication protocols and network infrastructures. The GreenCloud architecture is shown in Figure 4-3.

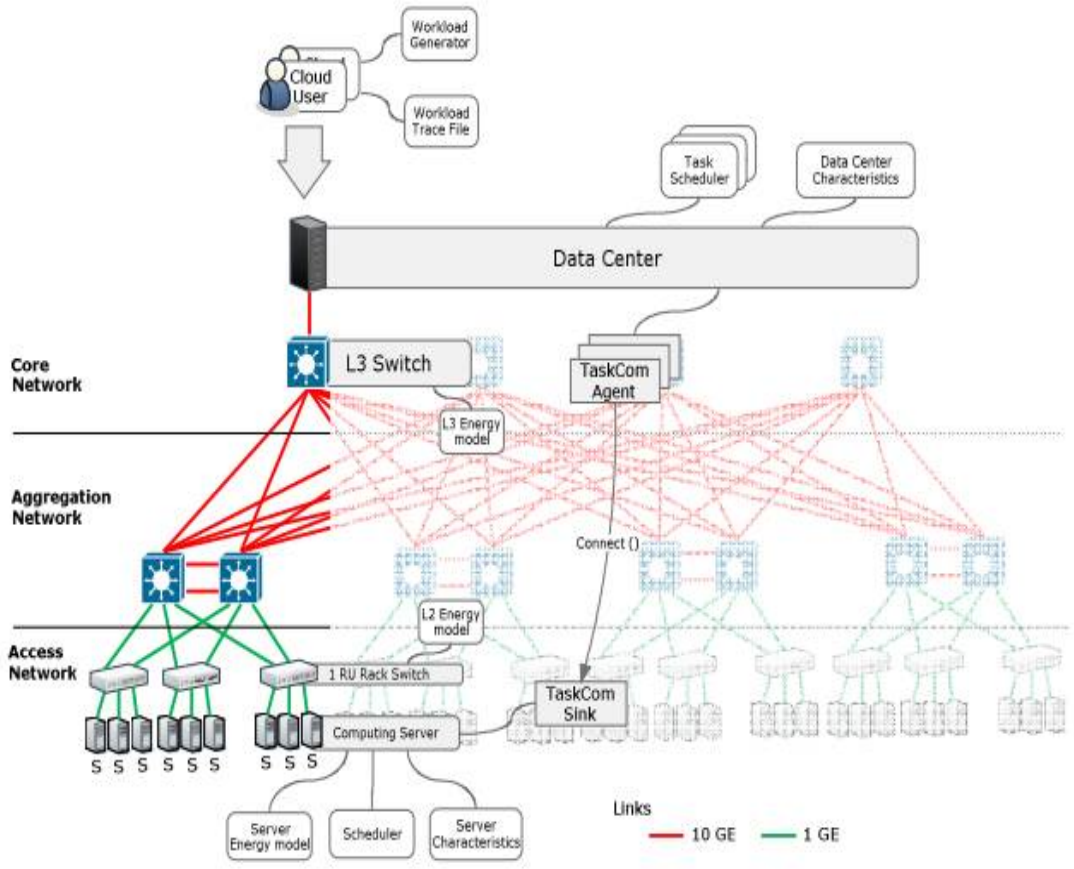


Figure 4-3: - Architecture of the GreenCloud Simulation Environment [26]

6. ***iCanCloud:*** - iCanCloud [31] is a cloud computing simulation platform capable of conducting large-scale experiments. This simulator is developed over SIMCAN (a simulation tool to analyze high performance I/O architectures) and uses the popular OMNET++ framework for simulating computer networks. It provides a scalable, flexible, fast, and easy to use tool that allows organizations to optimize the trade-offs between the cost and performance of their cloud computing system. The building block of a cloud computing system in this simulator is the virtual

machine. A virtual machine can be built by selecting and configuring its four main components: CPU, memory, storage, and network. Additionally, it provides a graphical interface that helps users in configuring their cloud computing experiments. It also, enables users to test different architectures simply by creating a new configuration file without modifying the simulator code. The configuration of a cloud computing system in iCanCloud consists of three main parts which are the cloud environment definition, the cloud hypervisor definition, and the user's configuration. The iCanCloud architecture is shown in Figure 4-4.

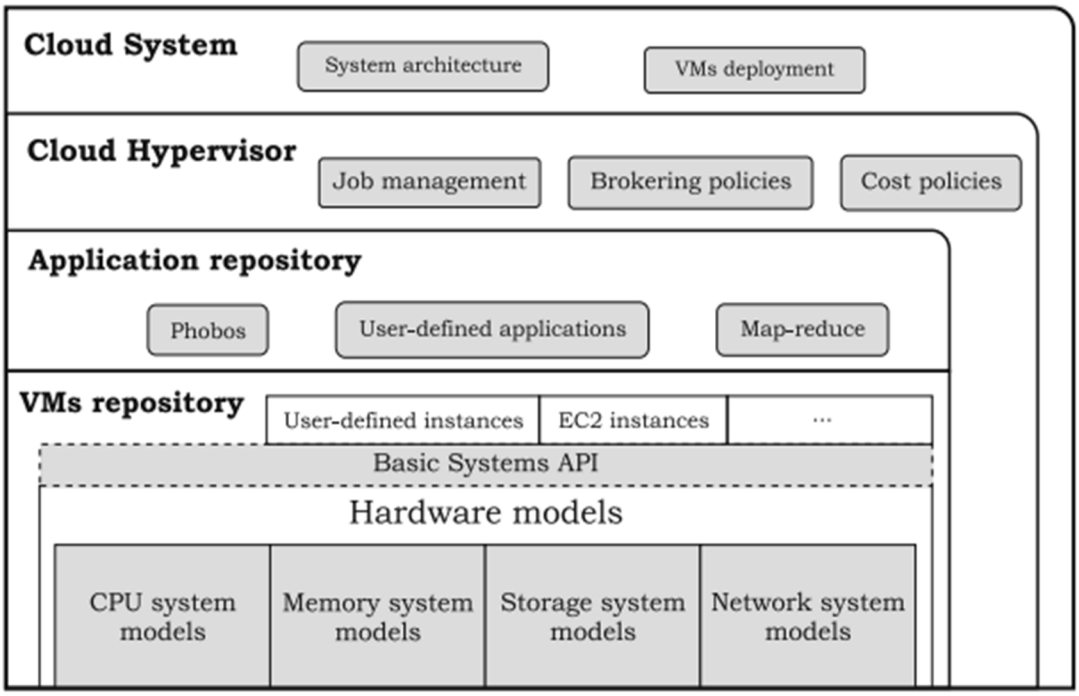


Figure 4-4: - Architecture of iCanCloud [31]

7. ***NetworkCloudsim:*** NetworkCloudSim is proposed by S.K. Garg and Rajkumari Buyya [27] and it is an extension of CloudSim. It is simulation framework that supports generalized applications such as high-performance computing applications, workflows, and e-commerce besides real cloud data centers modeling. It uses network topology class which implements network layer in CloudSim, reads a BRITE file and generates a topological network. It allows for modeling of Cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations. It structure supports designing of the real Cloud data centers and mapping different strategies. Information of network CloudSim is used to simulate latency in network traffic of CloudSim. The architecture of CloudSim-based NetworkCloudSim is depicted in Figure 4-5.

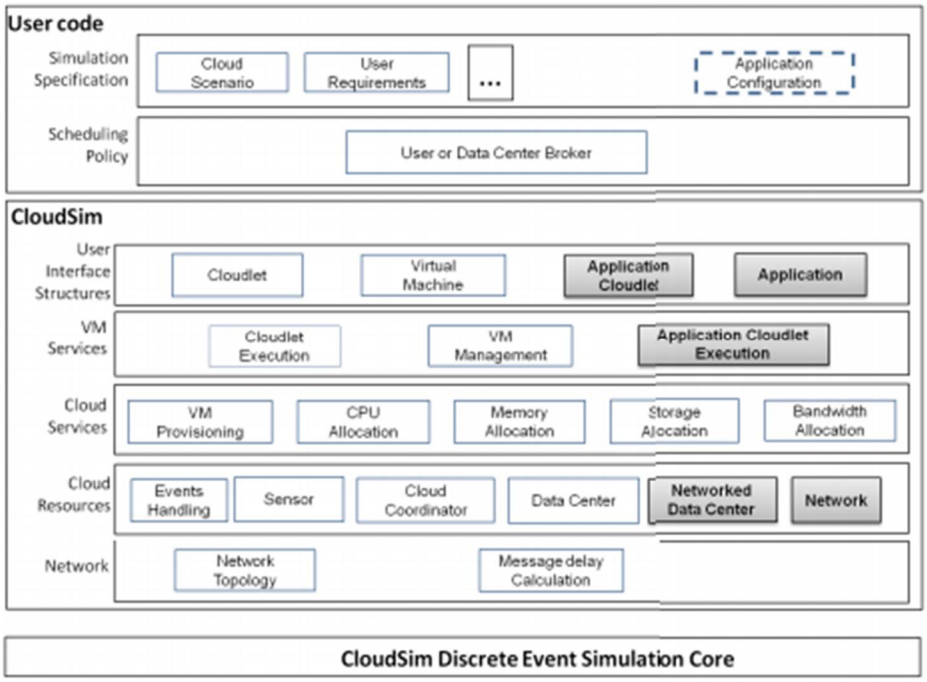


Figure 4-5: - The CloudSim Architecture with NetworkCloudSim Elements [27]

8. ***EMUSIM:*** EMUSIM [32] is an integrated architecture built on top of Automated Emulation Framework (AEF) for emulation and CloudSim for the simulation to anticipate service's behavior on cloud platforms to a higher Standard. It combines emulation and simulation to extract information automatically from the application behavior via emulation and uses this information to generate the corresponding simulation model. This is especially useful when the tester has no idea on the performance of the software under different levels of concurrency and parallelism, which impedes utilization of simulation. The architecture of EMUSIM is showed in Figure 4-6.

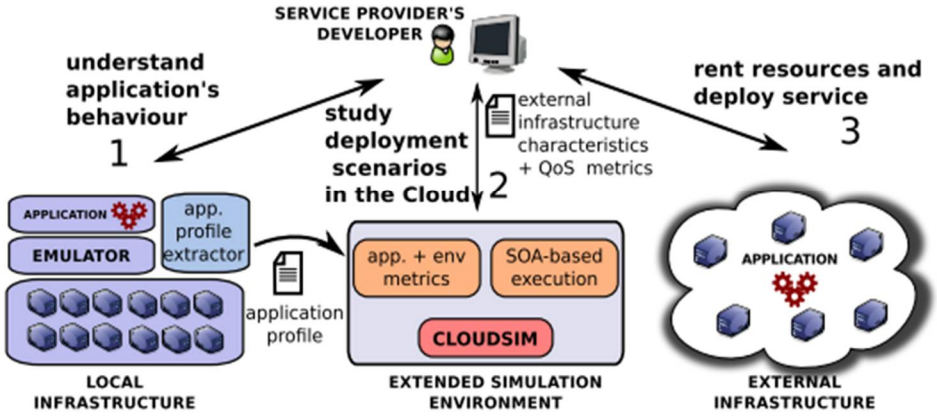


Figure 4-6: - EMUSIM Organization Overview [32]

Table 4-1: -Comparison of Cloud Computing Simulators

Cloud Simulators	***Parameters***						
	Platform	***GUI Support***	***Language***	***Networking***	***Software/ Hardware***	***Simulator Type***	***Availability***
CloudSim	GridSim	Limited	Java	limited	Software	Event Based	Open Source
CloudAnalyst	CloudSim	Full	java	limited	Software	Event Based	Open Source
CloudReport	CloudSim	Full	java	Limited	Software	Event Based	Open Source
CloudExp	CloudSim	Full	java	Full	Software	Event Based	No
iCanCloud	OMNET++, MPI	Full	C++	Full	Software	-	Open Source
GreenCloud	NS2	Limited	C++/ OTcl	Full	Software	Event Based	Open Source
NetworkCloud	CloudSim	Limited	java	Limited	Software	Event Based	Open Source
EMUSIM	AEF, CloudSim	Limited	java	Limited	Software	Event Based	Open Source

4.2.3.2.CloudSim

Based on the above discussion and comparison between cloud simulation tools (see Table 4-1). We chose CloudSim as our simulation tool to conduct this research. Because, CloudSim is highly scalable, reputable, flexible, and accurate than the other simulators.

CloudSim [25] is a multi-layered software framework written in java which enables seamless modeling, simulation, and experimentation of cloud computing and application services. The architecture of CloudSim is shown in Figure 4-7. It is developed to overcome the inability of distributed system simulators to evaluate the performance of cloud provisioning policies, services, application workload, models and resources performance models under the varying system, user configurations and requirements. It offers the following novel features [25]:

i. Supports for modeling and simulation of large-scale Cloud computing environments, including data centers, on a single physical computing node.
ii. A self-contained platform for modeling Clouds, service brokers, provisioning, and allocation policies.
iii. Support for simulation of network connections among the simulated system elements.
iv. Facility for simulation of federated Cloud environment that inter-networks resources from both private and public domains.

First releases of CloudSim used SimJava as the discrete event simulation engine that supports several core functionalities, such as queuing and processing of events, the creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation time. However, in the current release, the SimJava layer has been removed to allow some advanced operations that are not supported by it.

The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. The fundamental issues, such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state, are handled by this layer.

A Cloud provider, who wants to study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), would need to implement his strategies at this layer. Such implementation can be done by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer related to the provisioning of hosts to VMs. A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider's defined QoS levels.

This layer also exposes the functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study. The top-most layer in the CloudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies.

By extending the basic entities given at this layer, a Cloud application developer can perform the following activities:

i. Generate a mix of workload request distributions, application configurations.
ii. Model Cloud availability scenarios and perform robust tests based on the custom configurations.
iii. Implement custom application provisioning techniques for clouds and their federation.

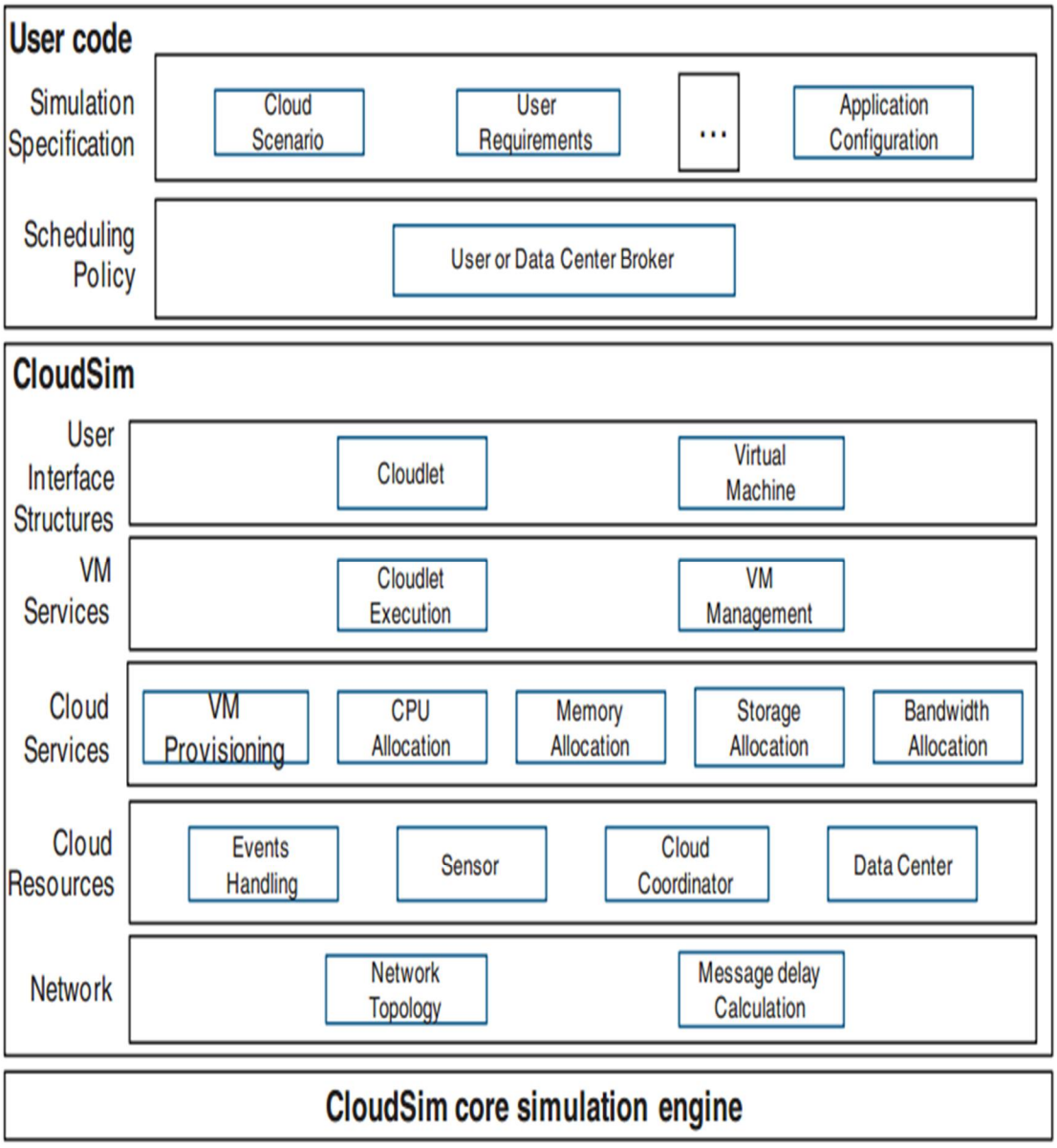


Figure 4-7: - Layered CloudSim Architecture [25]

4.2.3.2.1. Classes in CloudSim

The Class design diagram for the CloudSim is showed in Figure 4-8. In this section, we provide details associated with the fundamental classes of CloudSim, which are building blocks of the simulator [25].

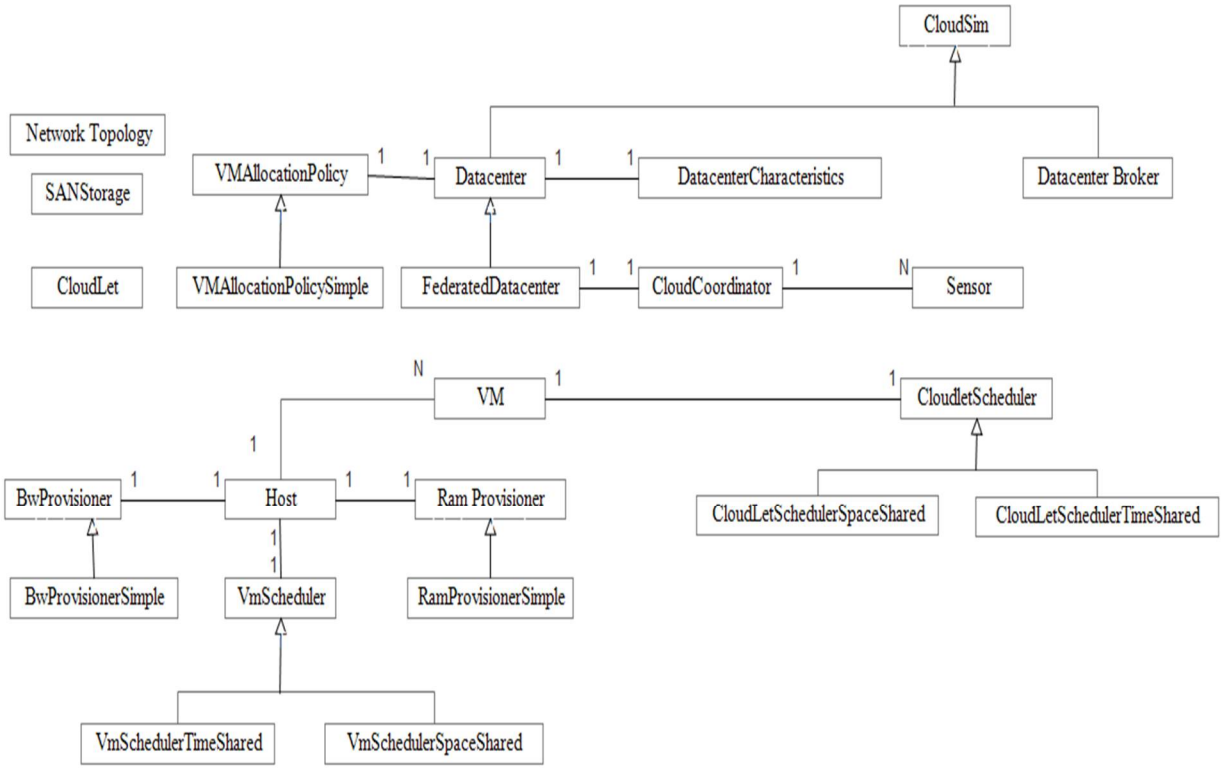


Figure 4-8: - CloudSim Class Design Diagram [25]

1. ***Datacenter:*** - It is a collection of computing hosts that can be either heterogeneous or homogeneous as regard to their resource configurations such as memory, capacity cores, and storage etc. Moreover, every datacenter instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices.

2. ***Datacenter Broker:*** - It is a class which is responsible for an intermediate between user and service providers which depend on user's QoS requirements and deploys service tasks across clouds. Cloud broker acts on behalf of cloud service providers. The system developer and researcher can extend this class for conducting experiments with their custom developed application placement policies.

3. ***VM:*** **-** This class makes models which are used to managed and hosted by cloud host components. Every VM components have access to a component which is extended from the abstract component called VM scheduling and these components can store the characteristics related to VM like memory, processor, VM'S internal provisioning policy storage etc.

4. ***Host:*** **-** It is a class which models physical resources like compute or storage server. It contains information like the amount of memory, storage, allocation of policy for sharing the processing power among VM's, list and type of processing power among VM's, policies for provisioning memory and bandwidth to the VM's.

5. ***Cloudlet:*** **-** It is a class which is used to make the cloud-based application services like social networking, content delivery and business workflow that are present in Datacenter. Cloudlet is also used to represent the complexity of any application in form of computational requirements.

6. ***Bandwidth Provisioner:*** **-** It is an abstract class that is used for provisioning of bandwidth to VM. The main task of this component is to undertake the allocation of network bandwidths to a set of running VMs that are hosted across the datacenter. If any researcher and developer need to extend this class, so they can extend this class with their own policies (priorities, QoS). BandwidthProvisoingSimple permits a VM to reserve as much as bandwidth as required. But, this is constrained by the total available bandwidth of the host.

7. ***Cloud Co-Coordinator:*** **-** It is abstract class extends a cloud-based datacenter to the federation. It is responsible for monitoring the internet state of datacenter resources by the updateDataCenter () method by detecting queries sensors.

8. ***Cloudlet Scheduler:*** **-** It is an abstract class extending through the implementation of different policies which are used to determining the share of processing powers among cloudlet in a VM. There are 2 types of provisioning policies. Time shared (CloudletSchedularTimeShared) and space shared (CloudletSchedularSpaceShared).

9. ***Datacenter Characteristics:*** **-** It is a class which is used to contain the configuration information of datacenter resources.

10. ***Network Topology:*** **-** It is a class which contains the important information for including network behavior in the simulation process and stores the topology information which is generated using the BRITE topology generator.

11. ***RAM Provisioner:*** **-** It is an abstract class which is used to represents the provisioning policy for allocating primary memory (RAM) to the VM's. The execution and development of VM on the host is feasible only when the RAM Provision components approve that the host has required ant of free memory.

12. ***SAN Storage:*** **-** It is a class which models a storage area network in the cloud-based datacenter for storing large chunks of data. San storage implements an interface which is used to simulate storage and retrieval of any ant of data, subject to the availability of network bandwidth.

13. ***VM Allocation Policy:*** **-**It is an abstract class which represents a provisioning policy. It means VM monitor are used for allocating VM to hosts. The main functionality of this class is to select any host which are available in a datacenter that meets the storage, memory, availability requirement for a VM.

14. ***VM Scheduler:*** **-** It is an abstract class which is implemented through host component that makes some policies like space shared and time shared. These policies are required for allocating the processing power to VM's.

4.2.4. Simulation Environment Configuration

In this section, we configured a cloud computing environment using the CloudSim toolkit. We create a datacenter which contains different hosts, VM, Cloudlets, and broker. This cloud environment is configured using the following parameters.

4.2.4.1.Datacenter Configuration

The experiment comprises only one datacenter which contains several hosts. There are two different type of hosts. These hosts are picked as they already used in CloudSim. Each type of hosts contains two hosts.

Table 4-2: - Configure Cloud Computing Datacenter

No.	***Total No. Host***	***System Architecture***	***OS***	***VMM***	***Cost***	***Cost per Memory***	***Cost per Bandwidth***	***Cost Per Storage***	***Time zone***
1	4	X86	Linux	Xen	3.0$	0.05$	0.0$	0.001$	10.0

Table 4-3: - Configure Hosts Parameters

No.	***No. Host***	***CPU core***	***MIPS***	***RAM***	***Storage***	***Bandwidth***	***Host Type***
1	2	2	2660	8192 MB	1 Tera	10GB	HP ProLiant ML110 G5
2	2	2	1860	8192 MB	1 Tera	10GB	HP ProLiant ML110 G4

4.2.4.2.Virtual Machine Configuration

For this experiment, we create four heterogeneous VM instances namely extra small, small, medium and large. Each VM instance comprises Two VMs.

Table 4-4: - Configure VM Parameters

No.	***No. of VM***	***CPU core***	***MIPS***	***RAM***	***Bandwidth***	***Instance Type***
1	2	1	2500	4096 MB	100MB	Large
2	2	1	2000	2048MB	100MB	Medium
3	2	1	1000	1024MB	1GB	small
4	2	1	500	512MB	1GB	Extra Small

4.2.4.3.Broker Configuration

In this section, we create a broker which works as an Interface between the cloud consumers and the cloud provider. The broker acts on behalf of the cloud user [25], it hides the creation, destruction, and management of VMs. For this Experiment, we only use one broker.

Table 4-5: - Configuration of Broker Parameters

No.	***No. of Broker***	***Broker ID***
1	1	Broker id

4.2.4.4.CloudLet Configuration

Each VM runs an application with the different workload. In CloudSim application, workloads are represented in the form of cloudlets. For this experiment, we use only one cloudlet. The cloudlet parameters are showed in the table below.

Table 4-6: - Configure CloudLet Parameters

No.	***No. of CloudLet***	***CPU core***	***Length***	***File Size***	***Output Size***
1	8	1	250000	300	300

4.3. Implementation

This section shows the implementation of the proposed approach. It begins by describing the basic steps to create a cloud computing environment in cloudSim toolkit. Then we will explain the implementation of the proposed VM migration algorithm. This algorithm is developed using java programming language in NetBeans IDE with the cloudSim toolkit. Finally, the performance metric used in this research paper and the experiment performed will be illustrated.

4.3.1. Steps for Creating a Basic Cloud Datacenter

There are eight elementary steps for creating and running the simulated cloud infrastructure using the CloudSim. Figure 4-9, shows these steps.

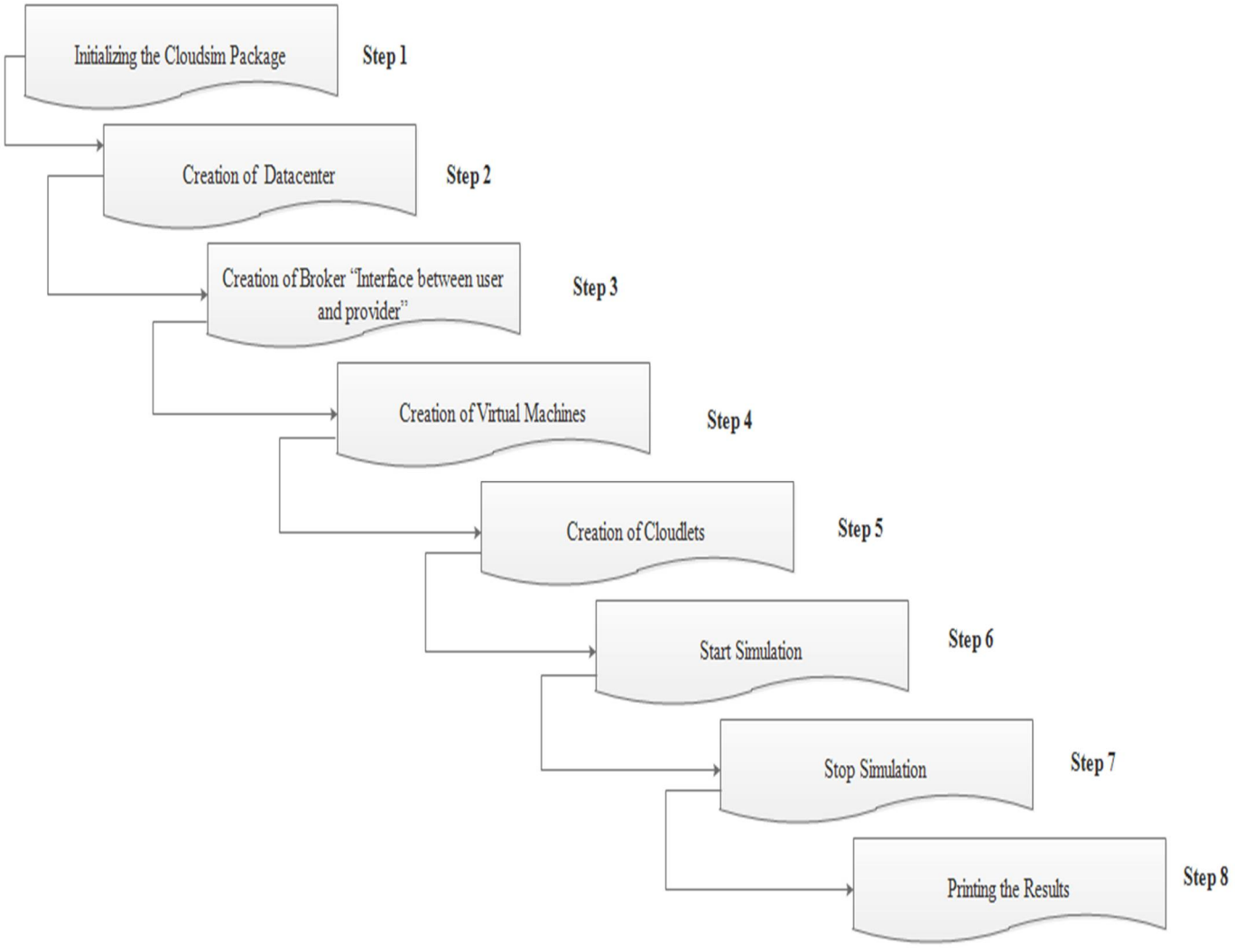


Figure 4-9: - Steps for creating a basic cloud

Step 1: - The first step is initializing the CloudSim package using the **init ()** method of the cloudSim class. It should be called before creating any entities. This initialization process includes three variables. The first variable represents the number of cloud user which denoted by the parameter **num_user**, the next one is the simulation starting time represented by the parameter **calendar** and the final parameter is **trace_flag** which traces the simulation events if set to true.

```
CloudSim.init (num_user, calendar, trace_flag);
```

Step 2: - The next step is creating one or more datacenters to run the cloudSim. The datacenter is a resource provider it contains several physical hosts. CloudSim uses createDatacenter ("datacenter_ name") method to create several datacenters. This method returns a datacenter object which comprises a list of hosts. Those hosts have their own computing resources like CPU, RAM, Bandwidth, and Storages.

```
Datacenter datacenter0 = createDatacenter("dataCentre_name");
Datacenter datacenter = new Datacenter (name, datacenterCharacteristics, VmAllocationPolicy,
storageList, schedulingInterval)
```

Where name is the name of the datacenter, datacenterCharacteristics is the characteristics of the datacenter, VmAllocationPolicy is the VM provision policy, storageList is the list of storages and schedulingInterval is the scheduling interval.

Step 3: - The third step is creating a broker using the createBroker () method which returns the datacenter broker object. This broker object will submit the VM to their corresponding hosts and the cloud lets to the VMs. The getId () method returns the broker id.

```
DatacenterBroker broker = createBroker();

int brokerId = broker.getId();
```

Step 4: - In this step, an array list object will be created to store the number of VMs created in the cloud datacenter as shown in the code bellow.

```
private static List<Vm> vmlist = new ArrayList<Vm> ();
```

After the VM list created, one or more VM object needs to be created using the VM class to be added to the VM list.

```
Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,
                    new CloudletSchedulerTimeShared());
```

Where vmid is the id of the VM, brokerId is the id of the broker object which is created in step three it is the owner of this VM object, mips refers to million instruction per second which maps to the CPUs frequency it is the VMs instruction execution rate, pesNumber is the number of CPU core the VM have, ram is the amount of memory in megabyte, bw is the network bandwidth in megabit, size is the size of the storage in megabyte, vmm is the type of hypervisor used and new CloudletSchedulerTimeShared() is the cloudlet scheduling policy. Finally, the list of VM is submitted to the broker for execution using the following method.

```
broker.submitVmList(vmlist);
```

.

Step 5: - In this step, a cloudlet object will create to run on the top of the VMs as the application services. A list of cloudlet will be created using the following line of code.

```
private static List<Cloudlet> cloudletList = new ArrayList<Cloudlet>();
```

After the cloudlet list is created, one or more cloudlet object needs to be created using the Cloudlet class to be added to the cloudlet list.

```
Cloudlet cloudlet = new Cloudlet (cloudletId, length, pesNumber, fileSize,
        outputSize, CPUutilizationModel, RAMutilizationModel, BWutilizationModel);
```

Where cloudletId is the id of the cloudlet object, length is the size of this cloudlet to be executed in MI (MIPS), pesNumber is the number of processing element (CPU core) to execute the cloudlet, fileSize is the input size of the cloudlet before execution in byte, outputSize is the output size of the cloudlet after the execution in byte, CPUutilizationModel is utilization of CPU model to monitor the cloudlet CPU usage, RAMutilizationModel is utilization of memory model to monitor the cloudlet memory usage and BWutilizationModel is utilization of bandwidth model to monitor the cloudlet network bandwidth usage. Finally, the list of cloudlet is submitted to the broker for execution using the following method.

```
broker.submitCloudletList(cloudletList)
```

Final, the VM and the cloudlet will be bind to execute together using the following code.

```
broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
```

Step 6: - After all entities, have been created, and the cloudlet is submitted to the VMs for the execution. The simulation will start using sartSimulation () method. This method waits for all the entities to finish the execution.

```
CloudSim.startSimulation();
```

Step 7: - The simulation will stop when the stopSimulation() method is invoked. This method will throw NullPointerException if any entity is going to be created before the initialization of the CloudSim.

```
CloudSim.stopSimulation();
```

Step 8: - Finally, the output messages and the values of the performance metrics are printed using the suitable methods. The proposed approach is built using those strategies stated in this section.

4.3.2. Integrating the Proposed Approach with CloudSim

Figure 4-10, shows the class hierarchy of the implementation of the proposed approach. This Diagram is constructed with the help of easyUML plugin which is plugged into the NetBeans IDE for Java developers. To implement the proposed approach, we create two different class. The first one is called Configuration class which is a subclass of Datacenter class that in turn is a child class of the SimEntity class in the org.cloudbus.cloudsim.core package and the other one is class NewBroker extends the DatacenterBroker class in the org.cloudbus.cloudsim package which is a subclass of SimEntity. Class SimEntity implements interface Cloneable. The proposed algorithm is hooked within class Configuration and invoked from class NewBroker using the send method of Class SimEntity to perform virtual machine migration between the source and target hosts.

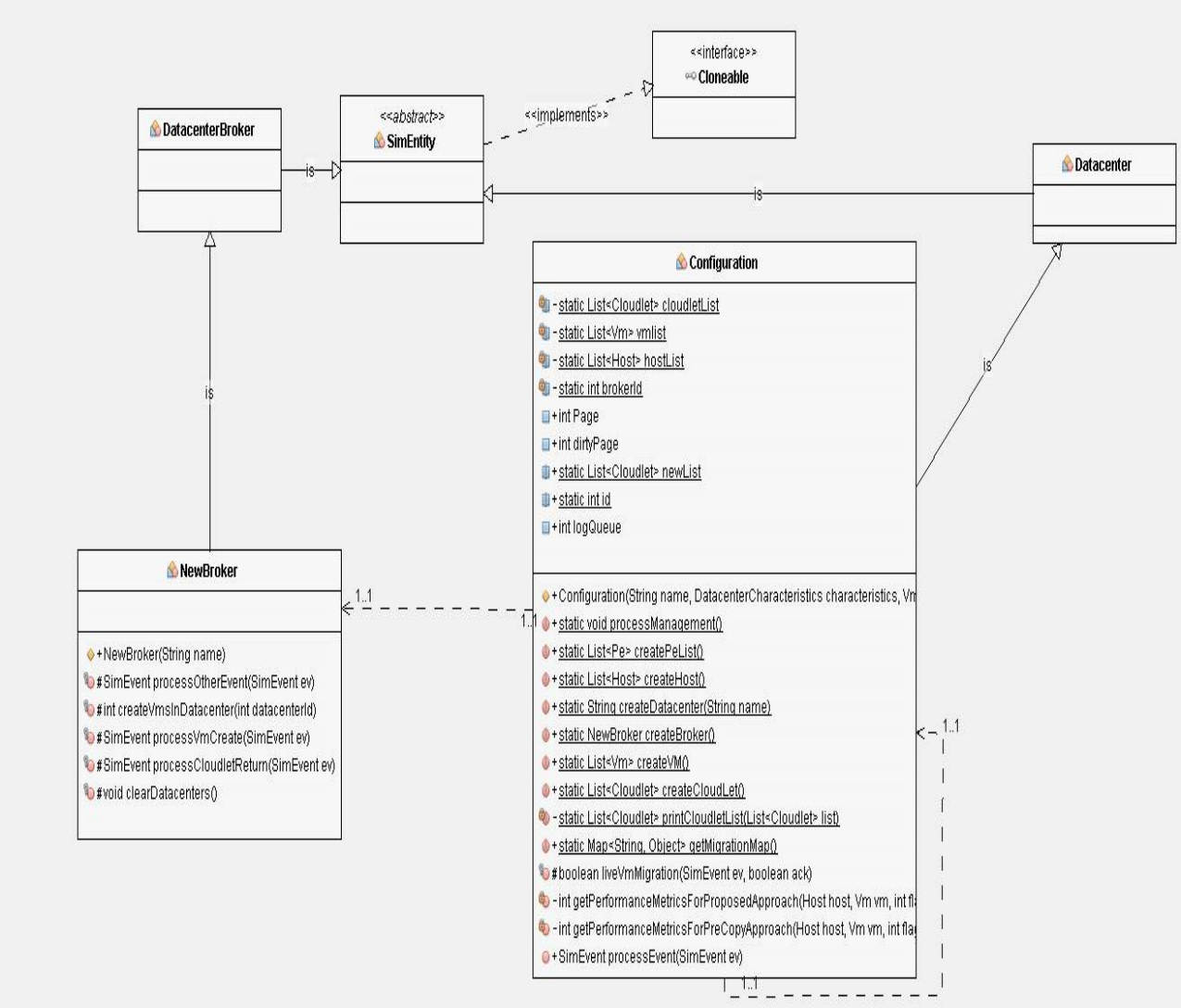


Figure 4-10: -The Class Hierarchy of the Proposed Approach

4.3.3. Performance Metrics

Performance metrics have been used for evaluation and comparison purposes to investigate the effectiveness of the proposed approach in comparison with other approaches. There are three main performance metrics used for evaluation in this paper which are:

Total Migration Time: **-** It is the period taken in finish transfer of virtual machine from source to destination. It should be small for fast completion of migration. Due to the transfer of additional pages other than the virtual machine memory size, migration time is also bigger. It is defined as the time taken by all round during migration.

Service Downtime: **-** It is the period taken by migration process to stop the virtual machine at the source and resume at the destination host. It directly affects the service availability and his value depends on the remaining pages in the last round. Downtime is measured as the time taken by the last iteration of the migration process.

Total Number of pages Transferred**:** **-** It is the total number of pages moved during migration. For healthier performance, this value should be minimum and ideally, it should be equivalent to the total number of pages of the virtual machine. The total pages moved is defined as the total number of pages in all round.

4.3.4. Experiments

This section defines the different experiments conducted for evaluation and comparison purposes. For this purpose, we perform three different experiments namely Experiment 1, 2 and 3 which is used to calculate total migration time, downtime and a total number of pages transferred in the proposed approach and pre-copy based live migration. Assuming that the memory could be processed in the proposed approach as pages with a fixed size equal to **P** bytes, **M** is the memory size, the bandwidth available for the transfer is represented by **B**, and P_{dirty} is the modified page per second. Details of the physical and virtual machines used in the experiments are described in Chapter 4 “Simulation Environment Configuration” Section 4.1.4. The whole experiments simulation result is presented in this section.

Table 4-7: - Annotation for Calculation Results

P	Page Size (4KB)
M	Memory Size (131072 page or 512 MB, 262144 page or 1024MB, 524288 page or 2048MB and 1048574 page or 4096MB)
B	Transfer Rate (5MB, 100MB, and 1GB per second for different connection)
P_{dirty}	Modified Pages per second (512 pages per second)
T_i	Execution time of i^{th} iteration for push phase
T_{stop}	Execution time for stop phase
T_{total}	Total Migration Time
T_{run}	Execution time for run and copy phase
Log_{queue}	Queue to store the whole CPU state (it contains pages less than 30)
P_{total}	Total number of page transfer in this migration process

Therefore, the time cost to transfer the memory pages and CPU state in the proposed approach is defined as follows:

The execution time of run and copy phase of the proposed approach:

$$T_{run} = \frac{M_1 P + Log_{queue}}{B} \quad \text{..... (1)}$$

Where $M_1 = M$ for first iteration and then modified memory page size is updated by:

$$M_2 = T_{run} P_{dirty} \quad \text{..... (2)}$$

Execution time for stop phase of proposed approach based live migration:

$$T_{stop} = \frac{M_2 P + Log_{queue}}{B} \quad \text{..... (3)}$$

$$T_{total} = T_{run} + T_{stop} \quad \text{..... (4)}$$

$$P_{total} = M_1 + M_2 \quad \text{..... (5)}$$

And the time cost to transfer the memory pages in pre-copy based live migration is as follows [3, 16, 17, 19, 21, 22]:

The execution time of i^{th} iteration for push phase of pre-copy based live migration:

$$T_i = \frac{M_i P}{B} \quad \text{.....(6)}$$

Where i=1,2, 3, …. n and $M_n \leq P_{dirty}$ and $M_1 = M$ for first iteration and then modified memory page size is updated by:

$$M_{i+1} = T_i P_{dirty} \quad \text{.....(7)}$$

"Where $M_{i+1} < B$ stop and copy, phase will start."

Execution time for stop phase of Pre-copy based live migration:

$$T_{stop} = \frac{M_n \ P}{B} \quad \text{..... (8)}$$

$$T_{total} = \sum_i^n T_i + T_{Stop} \quad \text{..... (9)}$$

$$P_{total} = \sum_{i=1}^n M_i \quad \text{..... (10)}$$

4.3.4.1. Experiment One

Objective: - To perceive the efficiency of live virtual machine migration algorithms, where migrating VM with a different size between two hosts within the same datacenter over 1 GB/s network bandwidth.

Configuration: For achieving the objectives of experiment one, four dissimilar configurations have been selected as follows:

- ***Configuration 1.1: -*** In the first configuration of experiment 1, the datacenter consists of 2 hosts and there are 2 virtual machines with the size of 512MB allocated to those hosts. Figure 4-11, shows the snapshot of the live migration and performance metrics that resulted from running the proposed approach over the 1GB/s network bandwidth.

  In the other hand, Figure 4-12, shows the snapshot result of the pre-copy approach over 1GB/s data transfer rate.

- ***Configuration 1.2: -*** The second configuration of experiment one is conducted by increasing the size of the virtual machine allocated to the hosts from 512MB to 1024MB and migrating this VM from one host to another host over 1GB/s network bandwidth. Figure 4-13, shows the snapshot result of the proposed approach and Figure 4-14, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty

                                        ========== OUTPUT OF PROPOSED APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

       0               1               2            0        512        1000         0.515026                0.0              131072                             131334.656                         2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-11: - Migrating VM with the Size of 512 MB Over 1 GB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty
500.1:totalTime:0.512
500.1:Next Memory:1.024
500.1:total Page:513.024

                                        ========== OUTPUT OF Pre-Copy APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

       0               1               2            0        512        1000         1.536                   1.024            131072                             131334.144                         2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-12: - Migrating VM with the Size of 512 MB Over 1 GB Network Bandwidth Using Pre-Copy Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
125.1: Broker: Cloudlet 0 received
125.1: Broker: Cloudlet 1 received
125.10: Migration of VM #0 from Host #0 to Host #1 is started
125.1: VM #0 has been deallocated from host #0
125.10: VM #0 has been allocated to the host #1
125.10: Migration of VM #0 to Host #1 is completed Successfuly
125.10: Host Id #1 contain VM Id #1
125.10: Host Id #1 contain VM Id #0
125.10: Host Id #0 is Empty
125.1:totalTime:2.040
125.1:Next Memory:4.096
125.1:total Page:2052.096
125.1:totalTime:2.052096
125.1:Next Memory:0.008192
125.1:total Page:2052.104192

                                        ========== OUTPUT OF Pre-Copy APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0               1               2            0        2048        1000    2.0602880000000003           0.008192               524288                          525338.673152               3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 125.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-13: - Migrating VM with the Size of 1024 MB Over 1 GB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty
500.1:totalTime:1.024
500.1:Next Memory:2.048
500.1:total Page:1026.048
500.1:totalTime:1.026048
500.1:Next Memory:0.004096
500.1:total Page:1026.052096

                                        ========== OUTPUT OF Pre-Copy APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0               1               2            0        1024        1000           1.0301440000000002           0.004096               262144                          262669.336576               3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-14: - Migrating VM with the Size of 1024 MB Over 1 GB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 1.3: -*** The third configuration of experiment one takes place by migrating the VM with the size of 2048MB over 1GB/s network bandwidth from one node to the other in the datacenter. Figure 4-15, shows the snapshot result of the proposed approach and Figure 4-16, shows the snapshot result of the pre-copy approach.

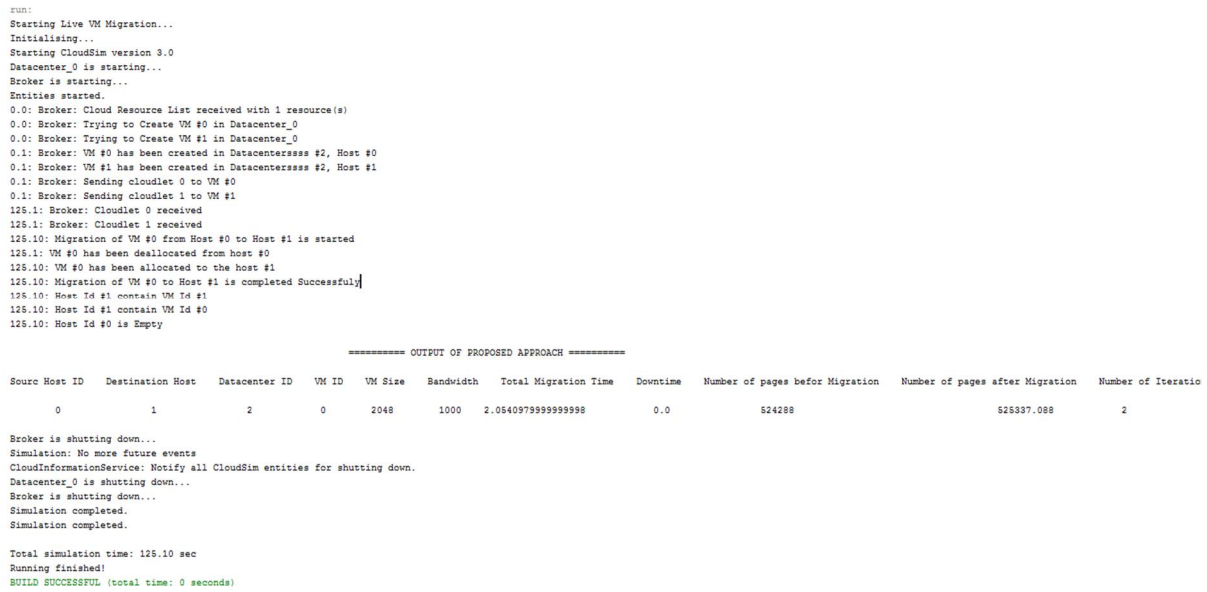


Figure 4-15: - Migrating VM with the Size of 2048 MB Over 1 GB Network Bandwidth Using the Proposed Approach

Figure 4-16: - Migrating VM with the Size of 2048 MB over 1 GB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 1.4: -*** The final configuration of experiment one is conducted by allocating VM with the of 4096MB to each physical machine in the datacenter and migrating the VM from one host to another over 1GB/s data transfer rate. Figure 4-17, shows the snapshot result of the proposed approach and Figure 4-18, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
100.1: Broker: Cloudlet 0 received
100.1: Broker: Cloudlet 1 received
100.10: Migration of VM #0 from Host #0 to Host #1 is started
100.1: VM #0 has been deallocated from host #0
100.10: VM #0 has been allocated to the host #1
100.10: Migration of VM #0 to Host #1 is completed Successfuly
100.10: Host Id #1 contain VM Id #1
100.10: Host Id #1 contain VM Id #0
100.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2            0        4096        1000        4.106194                0.0                1048576                            1050673.664                2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 100.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-17: - Migrating VM with the Size of 4096 MB Over 1 GB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
100.1: Broker: Cloudlet 0 received
100.1: Broker: Cloudlet 1 received
100.10: Migration of VM #0 from Host #0 to Host #1 is started
100.1: VM #0 has been deallocated from host #0
100.10: VM #0 has been allocated to the host #1
100.10: Migration of VM #0 to Host #1 is completed Successfuly
100.10: Host Id #1 contain VM Id #1
100.10: Host Id #1 contain VM Id #0
100.10: Host Id #0 is Empty
100.1:totalTime:4.096
100.1:Next Memory:8.192
100.1:total Page:4104.192
100.1:totalTime:4.104192
100.1:Next Memory:0.016384
100.1:total Page:4104.208384

========== OUTPUT OF Pre-Copy APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2            0        4096        1000    4.120576000000001           0.016384                1048576                            1050677.346304             3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 100.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-18: - Migrating VM with the Size of 4096 MB Over 1 GB Network Bandwidth Using Pre-Copy Approach

4.3.4.2. Experiment Two

Objective: To observe the efficiency of live virtual machine migration algorithms, where migrating VM with a different size between two hosts within the same datacenter over 100MB/s network bandwidth.

Configuration:

- ***Configuration 2.1:*** Migrating VM with the size of 512MB over 100MB/s between two hosts within the same datacenter. Figure 4-19, shows the snapshot result of the proposed approach and Figure 4-20, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2            0        512        100        5.2425999999999995        0.0              131072                              133698.56                  2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-19: - Migrating VM with the Size of 512 MB Over 100 MB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty
500.1:totalTime:5.12
500.1:Next Memory:10.24
500.1:total Page:522.24
500.1:totalTime:5.2224
500.1:Next Memory:0.2048
500.1:total Page:522.4448

========== OUTPUT OF Pre-Copy APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2            0        512        100              7.2704            2.048              131072                              133745.8688                3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-20: - Migrating VM with the Size of 512 MB Over 100 MB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 2.2:*** Migrating VM with the size of 1024MB over 100MB/s between two hosts in the same datacenter. Figure 4-21, shows the snapshot result of the proposed approach and Figure 4-22, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2            0        1024        100        10.465                  0.0              262144                             267392.0                           2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-21: - Migrating VM with the Size of 1024 MB Over 100 MB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty
500.1:totalTime:10.24
500.1:Next Memory:20.48
500.1:total Page:1044.48
500.1:totalTime:10.4448
500.1:Next Memory:0.4096
500.1:total Page:1044.8896

========== OUTPUT OF Pre-Copy APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2            0        1024        100          14.5408               4.096            262144                             267491.7376                        3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-22: - Migrating VM with the Size of 1024 MB Over 100 MB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 2.3:*** Migrating VM with the size of 2048MB over 100MB/s between two hosts in the same datacenter. Figure 4-23, shows the snapshot result of the proposed approach and Figure4-24, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
125.1: Broker: Cloudlet 0 received
125.1: Broker: Cloudlet 1 received
125.10: Migration of VM #0 from Host #0 to Host #1 is started
125.1: VM #0 has been deallocated from host #0
125.10: VM #0 has been allocated to the host #1
125.10: Migration of VM #0 to Host #1 is completed Successfuly
125.10: Host Id #1 contain VM Id #1
125.10: Host Id #1 contain VM Id #0
125.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2             0        2048        100        20.909799999999997          0.0                524288                              534778.88             2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 125.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-23: - Migrating VM with the Size of 2048 MB Over 100 MB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
125.1: Broker: Cloudlet 0 received
125.1: Broker: Cloudlet 1 received
125.10: Migration of VM #0 from Host #0 to Host #1 is started
125.1: VM #0 has been deallocated from host #0
125.10: VM #0 has been allocated to the host #1
125.10: Migration of VM #0 to Host #1 is completed Successfuly
125.10: Host Id #1 contain VM Id #1
125.10: Host Id #1 contain VM Id #0
125.10: Host Id #0 is Empty
125.1:totalTime:20.48
125.1:Next Memory:40.96
125.1:total Page:2088.96
125.1:totalTime:20.8896
125.1:Next Memory:0.8192
125.1:total Page:2089.7792

========== OUTPUT OF Pre-Copy APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2             0        2048        100            29.0816            8.192              524288                          534983.4752                    3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 125.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-24: - Migrating VM with the Size of 2048 MB Over 100 MB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 2.4:*** Migrating VM with the size of 4096MB over 100MB/s between two hosts in the same datacenter. Figure 4-25, shows the snapshot result of the proposed approach and Figure 4-26, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
100.1: Broker: Cloudlet 0 received
100.1: Broker: Cloudlet 1 received
100.10: Migration of VM #0 from Host #0 to Host #1 is started
100.1: VM #0 has been deallocated from host #0
100.10: VM #0 has been allocated to the host #1
100.10: Migration of VM #0 to Host #1 is completed Successfuly
100.10: Host Id #1 contain VM Id #1
100.10: Host Id #1 contain VM Id #0
100.10: Host Id #0 is Empty

                                          ========== OUTPUT OF PROPOSED APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2             0         4096         100          41.7994                0.0                1048576                          1069552.64                2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 100.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-25: -Migrating VM with the Size of 4096 MB Over 100 MB Network Bandwidth Using the Proposed Approach

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
100.1: Broker: Cloudlet 0 received
100.1: Broker: Cloudlet 1 received
100.10: Migration of VM #0 from Host #0 to Host #1 is started
100.1: VM #0 has been deallocated from host #0
100.10: VM #0 has been allocated to the host #1
100.10: Migration of VM #0 to Host #1 is completed Successfuly
100.10: Host Id #1 contain VM Id #1
100.10: Host Id #1 contain VM Id #0
100.10: Host Id #0 is Empty
100.1:totalTime:40.96
100.1:Next Memory:81.92
100.1:total Page:4177.92
100.1:totalTime:41.7792
100.1:Next Memory:1.6384
100.1:total Page:4179.5584

                                          ========== OUTPUT OF Pre-Copy APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2             0         4096         100          58.1632              16.384                1048576                          1069966.9504                3

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 100.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-26: - Migrating VM with the Size of 4096 MB Over 100 MB Network Bandwidth Using Pre-Copy Approach

4.3.4.3. Experiment Three

Objective: To see the efficiency of live virtual machine migration algorithms, where migrating VM with a different size between two hosts within the same datacenter over 5MB/s network bandwidth.

Configuration:

- ***Configuration 3.1:*** Migrating VM with the size of 512MB over 5MB/s between two hosts in the same datacenter. Figure 4-27, shows the snapshot result of the proposed approach and Figure4-28, shows the snapshot result of the pre-copy approach.

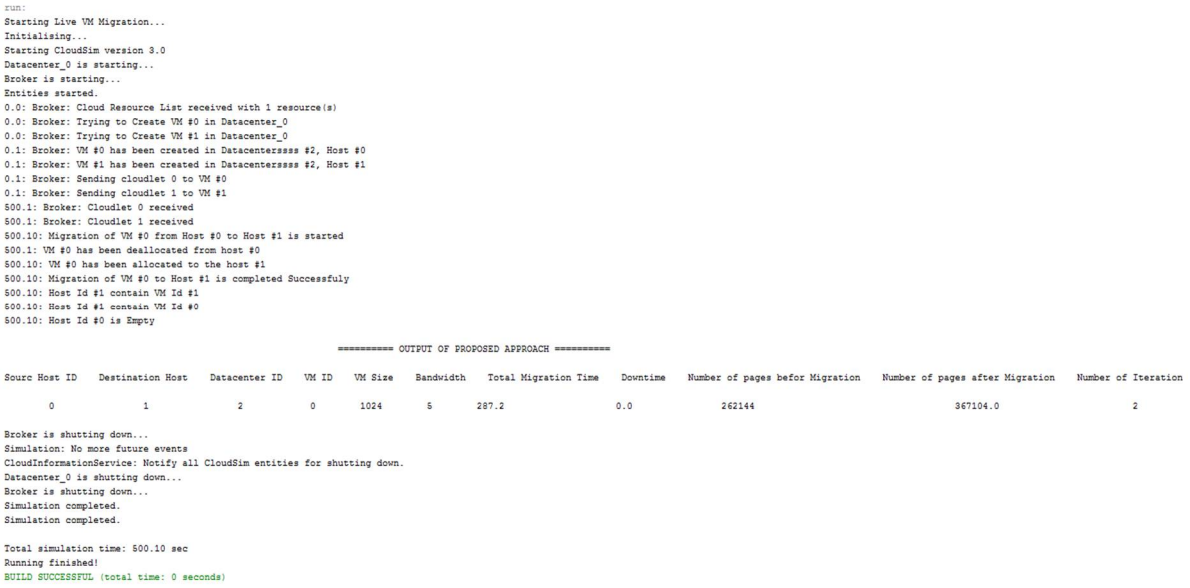
```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2             0        512         5        143.83999999999997        0.0              131072                              183603.2                    2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-27: - Migrating VM with the Size of 512 MB Over 5 MB Network Bandwidth Using the Proposed Approach

```
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty
500.1:totalTime:102.4
500.1:Next Memory:204.8
500.1:total Page:716.8
500.1:totalTime:143.36
500.1:Next Memory:81.92
500.1:total Page:798.7199999999999
500.1:totalTime:159.74400000000003
500.1:Next Memory:32.768
500.1:total Page:831.4879999999999
500.1:totalTime:166.29760000000002
500.1:Next Memory:13.1072
500.1:total Page:844.5952
500.1:totalTime:168.91904000000002
500.1:Next Memory:5.24288
500.1:total Page:849.83808
500.1:totalTime:169.96761600000002
500.1:Next Memory:2.097152000000003
500.1:total Page:851.935232
500.1:totalTime:170.38704640000003
500.1:Next Memory:0.8388608000000002
500.1:total Page:852.7740928000001

========== OUTPUT OF Pre-Copy APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2             0        512         5        338.1592064000001    167.77216000000004              131072                              218310.16775680002          8

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-28: -Migrating VM with the Size of 512 MB Over 5 MB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 3.2:*** Migrating VM with the size of 1024MB over 5MB/s between two hosts in the same datacenter. Figure 4-29, shows the snapshot result of the proposed approach and Figure 4-30, shows the snapshot result of the pre-copy approach.



```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
500.1: Broker: Cloudlet 0 received
500.1: Broker: Cloudlet 1 received
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2            0        1024         5         287.2                  0.0               262144                              367104.0                       2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-29: - Migrating VM with the Size of 1024 MB Over 5 MB Network Bandwidth Using the Proposed Approach

```
500.10: Migration of VM #0 from Host #0 to Host #1 is started
500.1: VM #0 has been deallocated from host #0
500.10: VM #0 has been allocated to the host #1
500.10: Migration of VM #0 to Host #1 is completed Successfuly
500.10: Host Id #1 contain VM Id #1
500.10: Host Id #1 contain VM Id #0
500.10: Host Id #0 is Empty
500.1:totalTime:204.8
500.1:Next Memory:409.6
500.1:total Page:1433.6
500.1:totalTime:286.72
500.1:Next Memory:163.84
500.1:total Page:1597.4399999999998
500.1:totalTime:319.48800000000006
500.1:Next Memory:65.536
500.1:total Page:1662.9759999999999
500.1:totalTime:332.59520000000003
500.1:Next Memory:26.2144
500.1:total Page:1689.1904
500.1:totalTime:337.83808000000005
500.1:Next Memory:10.48576
500.1:total Page:1699.67616
500.1:totalTime:339.93523200000004
500.1:Next Memory:4.194304000000001
500.1:total Page:1703.870464
500.1:totalTime:340.77409280000006
500.1:Next Memory:1.6777216000000004
500.1:total Page:1705.5481856000001

========== OUTPUT OF Pre-Copy APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2            0        1024         5          676.3184128000001    335.5443200000001          262144                   436620.33551360003              8

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 500.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-30: - Migrating VM with the Size of 1024 MB Over 5 MB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 3.3:*** Migrating VM with the size of 2048MB over 5MB/s between two hosts in the same datacenter. Figure 4-31, shows the snapshot result of the proposed approach and Figure4-32, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
125.1: Broker: Cloudlet 0 received
125.1: Broker: Cloudlet 1 received
125.10: Migration of VM #0 from Host #0 to Host #1 is started
125.1: VM #0 has been deallocated from host #0
125.10: VM #0 has been allocated to the host #1
125.10: Migration of VM #0 to Host #1 is completed Successfuly
125.10: Host Id #1 contain VM Id #1
125.10: Host Id #1 contain VM Id #0
125.10: Host Id #0 is Empty

========== OUTPUT OF PROPOSED APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2             0        2048         5         573.9200000000001          0.0               524288                          734105.6              2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 125.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-31: - Migrating VM with the Size of 2048 MB Over 5 MB Network Bandwidth Using the proposed Approach

```
125.10: Migration of VM #0 to Host #1 is completed Successfuly
125.10: Host Id #1 contain VM Id #1
125.10: Host Id #1 contain VM Id #0
125.10: Host Id #0 is Empty
125.1:totalTime:409.6
125.1:Next Memory:819.2
125.1:total Page:2867.2
125.1:totalTime:573.44
125.1:Next Memory:327.68
125.1:total Page:3194.8799999999997
125.1:totalTime:638.9760000000001
125.1:Next Memory:131.072
125.1:total Page:3325.9519999999998
125.1:totalTime:665.1904000000001
125.1:Next Memory:52.4288
125.1:total Page:3378.3808
125.1:totalTime:675.6761600000001
125.1:Next Memory:20.97152
125.1:total Page:3399.35232
125.1:totalTime:679.8704640000001
125.1:Next Memory:8.388608000000001
125.1:total Page:3407.740928
125.1:totalTime:681.5481856000001
125.1:Next Memory:3.3554432000000007
125.1:total Page:3411.0963712000002
125.1:totalTime:682.2192742400001
125.1:Next Memory:1.3421772800000003
125.1:total Page:3412.4385484800005

========== OUTPUT OF Pre-Copy APPROACH ==========
Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration
        0                1                2             0        2048         5          950.6547302400002    268.43545600000004              524288                    873584.2684108801                9

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 125.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-32: - Migrating VM with the Size of 2048 MB Over 5 MB Network Bandwidth Using Pre-Copy Approach

- ***Configuration 3.4:*** Migrating VM with the size of 4096MB over 5MB/s between two hosts in the same datacenter. Figure 4-33, shows the snapshot result of the proposed approach and Figure 4-34, shows the snapshot result of the pre-copy approach.

```
run:
Starting Live VM Migration...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenterssss #2, Host #0
0.1: Broker: VM #1 has been created in Datacenterssss #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
100.1: Broker: Cloudlet 0 received
100.1: Broker: Cloudlet 1 received
100.10: Migration of VM #0 from Host #0 to Host #1 is started
100.1: VM #0 has been deallocated from host #0
100.10: VM #0 has been allocated to the host #1
100.10: Migration of VM #0 to Host #1 is completed Successfuly
100.10: Host Id #1 contain VM Id #1
100.10: Host Id #1 contain VM Id #0
100.10: Host Id #0 is Empty

                                          ========== OUTPUT OF PROPOSED APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2             0        4096        5                1147.36                 0.0              1048576                            1468108.8                2

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 100.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-33: - Migrating VM with the Size of 4096 MB Over 5 MB Network Bandwidth Using the Proposed Approach

```
100.10: Host Id #0 is Empty
100.1:totalTime:819.2
100.1:Next Memory:1638.4
100.1:total Page:5734.4
100.1:totalTime:1146.88
100.1:Next Memory:655.36
100.1:total Page:6389.759999999999
100.1:totalTime:1277.9520000000002
100.1:Next Memory:262.144
100.1:total Page:6651.903999999995
100.1:totalTime:1330.3808000000001
100.1:Next Memory:104.8576
100.1:total Page:6756.7616
100.1:totalTime:1351.3523200000002
100.1:Next Memory:41.94304
100.1:total Page:6798.70464
100.1:totalTime:1359.7409280000002
100.1:Next Memory:16.777216000000003
100.1:total Page:6815.481856
100.1:totalTime:1363.0963712000002
100.1:Next Memory:6.7108864000000015
100.1:total Page:6822.1927424000005
100.1:totalTime:1364.4385484800002
100.1:Next Memory:2.6843545600000005
100.1:total Page:6824.877096960001
100.1:totalTime:1364.9754193920003
100.1:Next Memory:1.073741824000003
100.1:total Page:6825.950838784001

                                          ========== OUTPUT OF Pre-Copy APPROACH ==========

Sourc Host ID    Destination Host    Datacenter ID    VM ID    VM Size    Bandwidth    Total Migration Time    Downtime    Number of pages befor Migration    Number of pages after Migration    Number of Iteration

        0                1                2             0        4096        5             1579.7237841920003    214.74836480000008    1048576                 1747443.4147287041                 10

Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

Total simulation time: 100.10 sec
Running finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-34: - Migrating VM with the Size of 4096 MB Over 5 MB Network Bandwidth Using Pre-Copy Approach

5. CHAPTER FIVE: RESULT ANALYSIS AND EVALUATION

In this chapter, we discussed and analyzed the results we get from the 24 different simulation experiments conducted in cloudSim [25] for both the pre-copy approach and the new proposed approach. Those simulations are grouped into three categories as described in chapter 4 “Experiments” section 4.2.4. Additionally, we evaluated and compared the two approaches.

5.1. Result Analysis and Discussion

5.1.1. Total Migration Time

Total migration time is the time taken from the beginning to the end of the VM migration process. The proposed approach reduces the number of pages transferred because it migrates the whole VM only within two rounds. Due to this, total migration time is reduced.

Table 5-1 and Figure 5-1 shows the total migration time taken by both pre-copy approach and the proposed approach in 4.2.4.1. Experiment 1. When we migrate 512 MB VM over 1GM/s fast network bandwidth the total migration time taken by pre-copy approach is 1.5 second and the proposed approach is 0.5 second. The proposed approach reduced the total migration time by 66.7%. When the size of the VM increase from 512 MB to 1024 MB and migrate over 1 GB/s fast network the proposed approach reduced total migration time by 0.97%. the proposed approach also, reduced total migration time by 0.045% when 2048 MB VM is migrated over 1 GB/s fast network. When 4096 MB VM is migrated over 1 GB/s fast network the proposed approach reduced total migration time by 0.5% compared to the pre-copy approach.

The pre-copy approach number of iteration is reduced because of the 1 GB/s fast network we used to migrate the VM from source to the target host. Due to this the number of rounds of both the proposed approach and pre-copy approach almost similar and small total migration time difference.

Table 5-1: - Total Migration Time for Migrating Different VM Over 1 GB/s Network Bandwidth

VM Memory size	***Bandwidth***	***Pre-copy approach***	***Proposed approach***	***Reduction ratio (%)***
512 MB	1 GB/s	1.5 second	0.5 second	66.700
1024 MB	1 GB/s	1.03 second	1.02 second	0.970
2048 MB	1 GB/s	2.06 second	2.05 second	0.045
4096 MB	1 GB/s	4.12 second	4.10 second	0.500

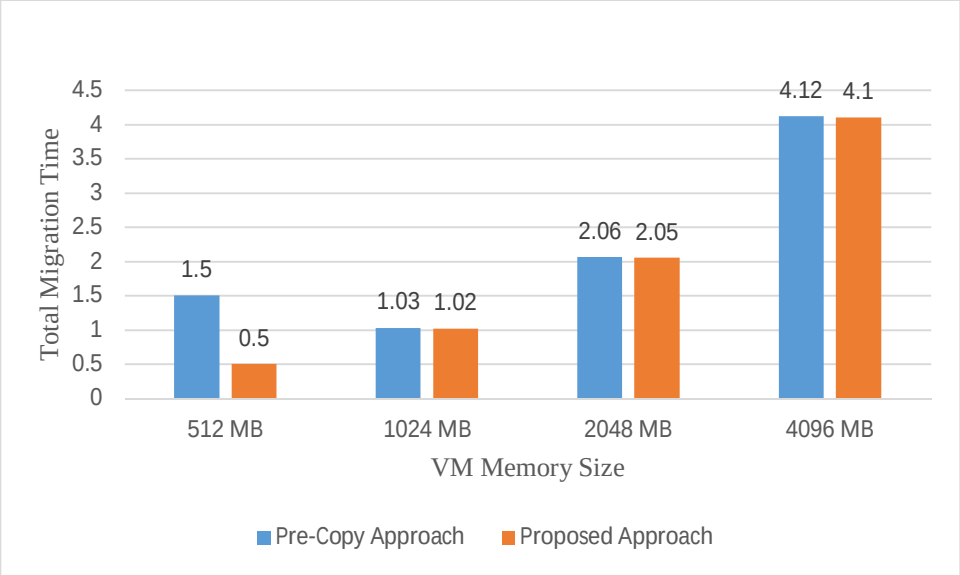


Figure 5-1: - Total Migration Time for Migrating Different VM Using Pre-Copy Approach and the Proposed Approach Over 1 GB Network Bandwidth

In 4.2.4.2. Experiment 2 the total migration time taken for both pre-copy and proposed approach is represented in Table 5-2 and Figure 5-2. The proposed approach reduced the total migration time when 512MB VM is migrated over 100MB/s bandwidth by 27.9%, 1024MB VM migrated over 100MB/s bandwidth by 27.86%, 2048MB VM is migrated over 100MB/s bandwidth by 28.129% and 4096MB VM is migrated over 100MB/s bandwidth by 28.146% as compared to the pre-copy approach.

Table 5-2: - Total migration time for migrating different VM over 100 MB/s network bandwidths

VM Memory size	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
512 MB	100 MB/s	7.27 second	5.24 second	27.900
1024 MB	100 MB/s	14.5 second	10.46 second	27.860
2048 MB	100 MB/s	29.08 second	20.9 second	28.129
4096 MB	100 MB/s	58.16 second	41.79 second	28.146

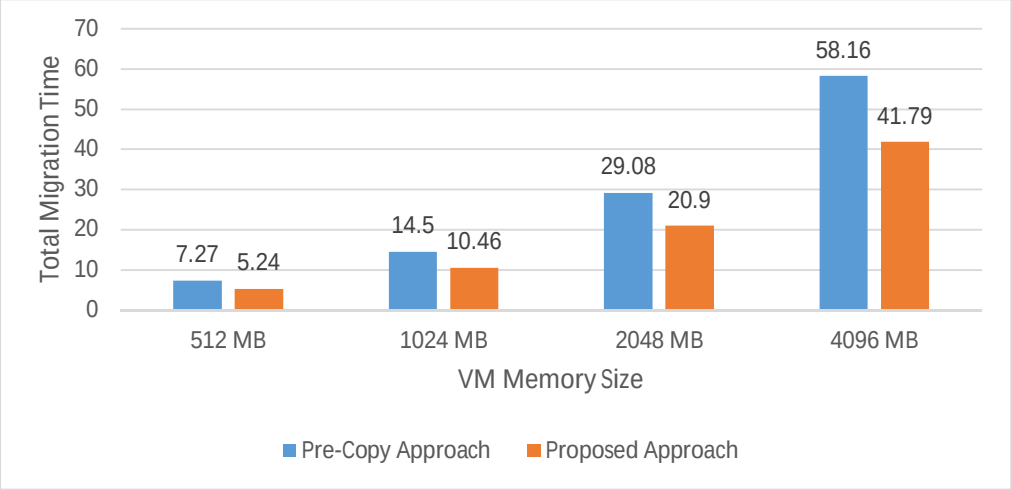


Figure 5-2: -Total Migration Time for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 100 MB/S Network Bandwidths

Migrating a VM over a slow network using a pre-copy approach will result with high total migration time compared to the proposed approach. The proposed approach always has only two iterations to migrate a VM but in the case of the pre-copy approach, the number of iteration to migrate the VM will increase when a large size of VM is migrated over a slow network. Because of this large number of iteration, the total migration time is increased.

Table 5-3 and Figure 5-3 shows the result of 4.2.4.3. Experiment 3 for both pre-copy approach and proposed approach. The total migration time taken by the pre-copy approach to migrate 512MB VM over 5MB/s slow network is 338.15 second but the proposed approach takes 143.8 seconds. This means the proposed approach reduced total migration time by 57.47%. total migration time taken to migrating the VM with the size of 1024MB over 5MB/s slow network using the pre-copy approach is 676.3 second but for the proposed approach is 287.2 second so, the proposed approach reduced the total migration time by 57.53%. Migrating a 2048MB VM over 5MB/s slow network using the pre-copy approach takes 950.65 second of total migration time but the total migration time taken to migrate this VM using proposed approach is 573.9 second. This shows the proposed approach reduced the total migration time by 39%. When the size of the VM increased into 4096MB and transferred over 5MB/s network the total migration time occupied by pre-copy approach is 1579.7 second and proposed approach is 1147.16 second. Once more the proposed approach reduced total migration time by 27%.

Finally, this expresses when the size of the VM increases and the network bandwidth used to transfer the VM reduced the number of iteration will increases this leads to big total migration time.

Table 5-3: - Total Migration Time for Migrating Different VM Over 5 MB/S Network Bandwidths

VM Memory size	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
512 MB	5 MB/s	338.15 second	143.8 second	57.47
1024 MB	5 MB/s	676.3 second	287.2 second	57.53
2048 MB	5 MB/s	950.65 second	573.9 second	39.60
4096 MB	5 MB/s	1579.7 second	1147.16 second	27.38

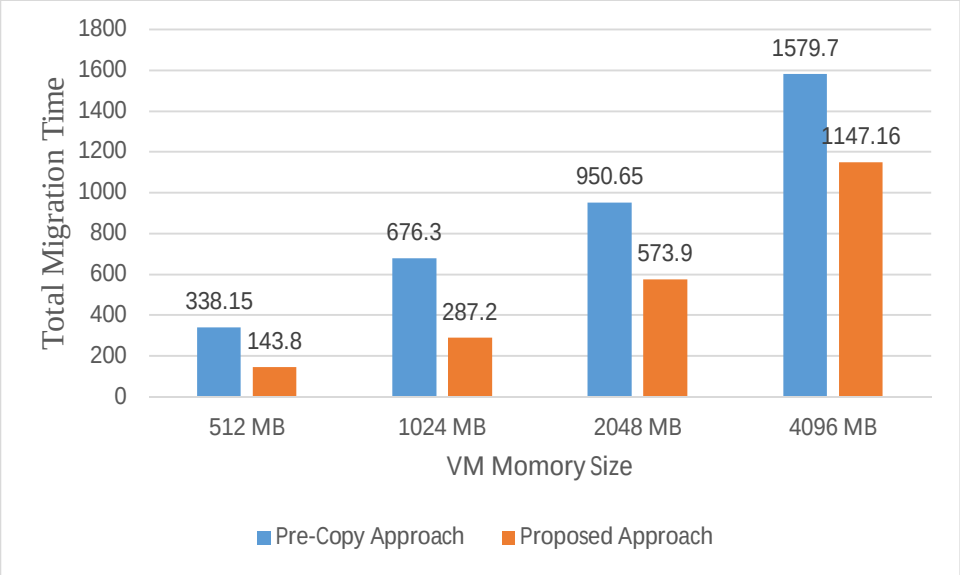


Figure 5-3: -Total Migration Time for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 5 MB/S Network Bandwidths

5.1.2. Total Page Transferred

Total page transferred is the number of pages copied during VM migration process. When VM is under low load or small size case, fewer pages are modified and most of the pages copied only once but for the case of heavy load or large size, many pages get modified and transfer frequently.

Table 5-4 and Figure 5-4 shows the total pages transferred obtained by both pre-copy approach and proposed approach from 4.2.4.1. Experiment 1. When we migrate 512MB VM over 1GB/s fast network the total pages transferred by pre-copy approach is smaller than the proposed approach by 0.0004%. this occurred because of the addition size of the logQueue which used to store the whole CPU state during run and copy phase of the proposed approach. But in the other case the proposed approach reduced the total pages transferred by 0.0002% when a VM with the size of 1024MB is migrated over 1GB/s network, 0.0003% when a VM with the size of 2048MB is migrated over 1GB/s network and 0.0004% when a VM with the size of 4096MB is migrated over 1GB/s network compared to the pre-copy approach.

Table 5-4: - Total Page Transferred for Migrating Different VM Over 1 GB Network Bandwidth

VM Memory size	*Bandwidth*	*Pre-copy*	*Proposed approach*	*Reduction ratio (%)*
512 MB	1 GB/s	131334.1	131334.6	-0.0004
1024 MB	1 GB/s	262669.3	262668.8	0.0002
2048 MB	1 GB/s	525338.6	525337.08	0.0003
4096 MB	1 GB/s	1050677.3	1050673.6	0.0004

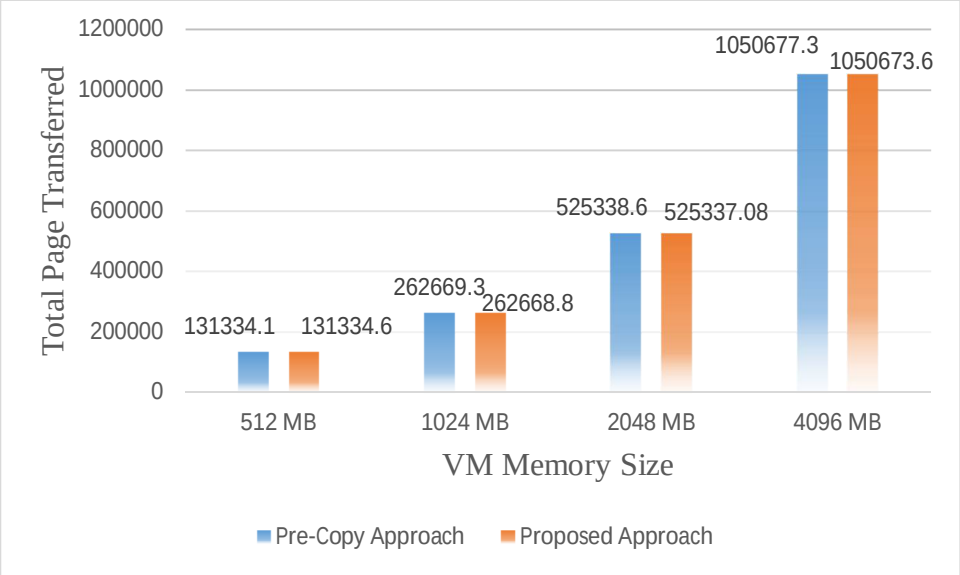


Figure 5-4: -Total Page Transferred for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 1 GB Network Bandwidth

In 4.2.4.2. Experiment 2 the total pages transferred resulted by both pre-copy and proposed approach is represented in Table 5-5 and Figure 5-5. The proposed approach reduced the total pages transferred when 512MB VM is migrated over 100MB/s bandwidth by 0.035%, 1024MB VM migrated over 100MB/s bandwidth by 0.037%, 2048MB VM is migrated over 100MB/s bandwidth by 0.038% and 4096MB VM is migrated over 100MB/s bandwidth by 0.0387% as compared to the pre-copy approach.

Table 5-5: - Total Page Transferred for Migrating Different VM Over 100 MB/S Network Bandwidths

VM Memory size	*Bandwidth*	*Pre-copy*	*Proposed approach*	*Reduction ratio (%)*
512 MB	100 MB/s	133745.86	133698.6	0.0350
1024 MB	100 MB/s	267491.7	267392	0.0370
2048 MB	100 MB/s	534983.47	534778.88	0.0380
4096 MB	100 MB/s	1069966.95	1069552.6	0.0387

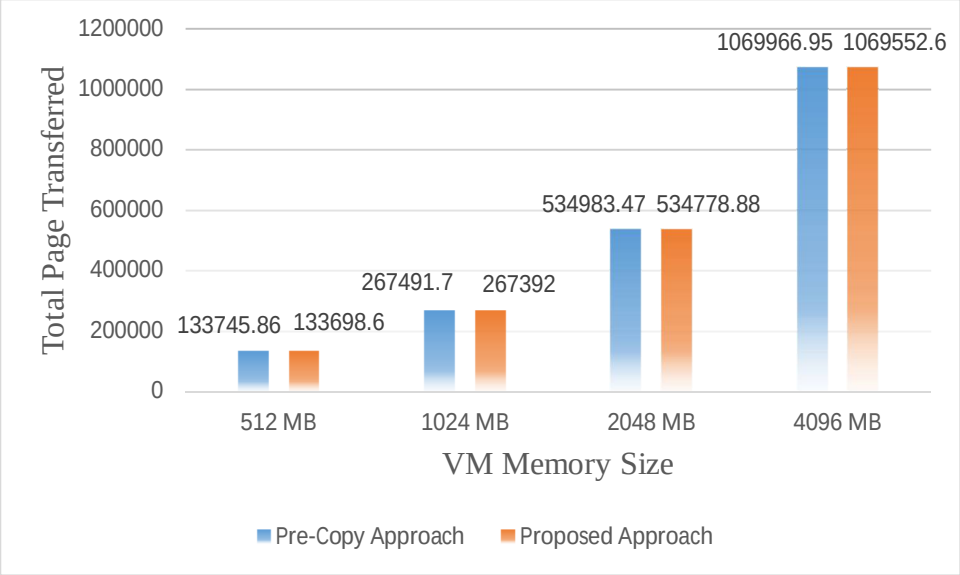


Figure 5-5: - Total Page Transferred for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 100 MB/S Network Bandwidths

Migrating dissimilar size of VM over the slow network using pre-copy approach and proposed approach had a different number of iteration this results with different total page transferred. The proposed approach reduced total pages transferred as compared to the pre-copy approach by 15.89% from migrating 512MB VM over 5MB slow network, 15.92% from migrating 1024MB VM over 5MB slow network, 15.96% from migrating 2048MB VM over 5MB slow network and 15.98% from migrating 4096MB VM over 5MB slow network as shown in Table 5-6 and Figure 5-6 gained from 4.2.4.3 Experiment 3.

Finally, the total pages transferred is increased in both pre-copy approach and proposed approach as the size of the VM is increased and the network bandwidth is decreased.

Table 5-6: -Total Page Transferred for Migrating Different VM Over 5 MB/S Network Bandwidths

VM Memory size	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
512 MB	5 MB/s	218310.16	183603.2	15.89
1024 MB	5 MB/s	436620.3	367104	15.92
2048 MB	5 MB/s	873584.26	734105.6	15.96
4096 MB	5 MB/s	1747443.4	1468108	15.98

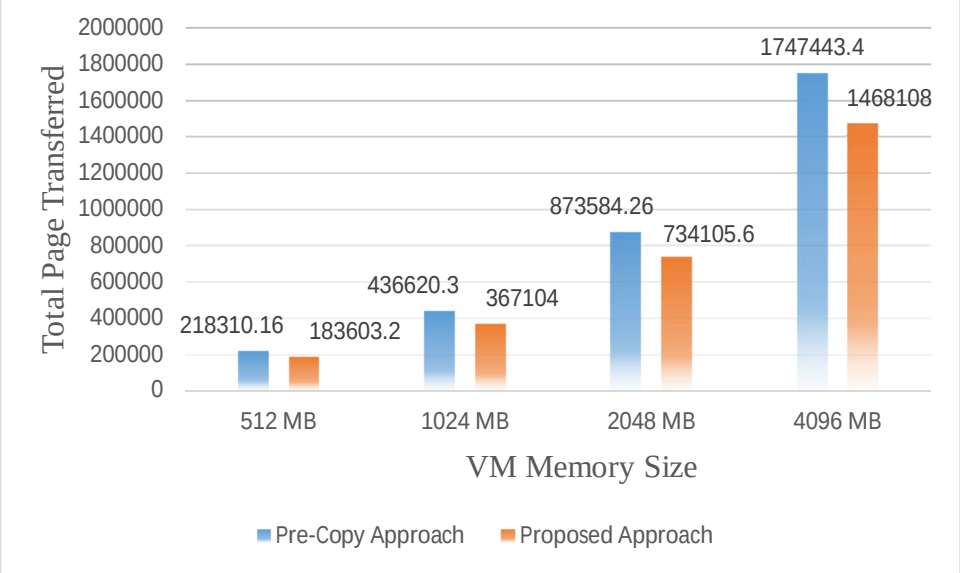


Figure 5-6: - Total Page Transferred for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 5 MB/S Network Bandwidths

5.1.3. Downtime

Service downtime is the period taken when the VM is stopped at source host and resumed at destination host after transferring remaining memory pages and CPU states. It is one of the significant parameters which directly affects the service availability. Its value should be as minimum as possible.

The proposed approach doesn't have the service downtime because it migrates the VM CPU state without interrupting the virtual machine running on the source host as described in chapter 3 "CPU-Register-State Migration" section 3.2.2. The proposed approach doesn't suspend the VM to copy its CPU state from source to the target node. Instead, it copies the VM CPU state while the VM is executing this eliminate the occurrence of service downtime. The stop and copy phase of the proposed approach only transfers the remaining pages of the VM. But in the case of the pre-copy [3] approach, the VM is suspended for some time to transfer the CPU state and resume at target host using the stop and copy phase as shown in Figure 2-11.

The proposed approach reduced the service downtime by 100% as compared to the pre-copy approach as shown in Table 5-7 and Figure 5-7 gained from 4.2.4.1. Experiment 1, Table 5-8 and Figure 5-8 obtained from 4.2.4.2. Experiment 2 and Table 5-9 and Figure 5-**9** gotten from 4.2.4.3. Experiment 3. In a pre-copy approach, the downtime is increased when the size of virtual machine increased and the network bandwidth used to transfer decreased but, in the case of the proposed approach, the service

downtime is 0 it doesn't matter whether the size of the VM increased or decreased and whether the network bandwidth increased or decreased.

Table 5-7: - Downtime for Migrating Different VM Over 1 GB Network Bandwidth

VM Memory size	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
512 MB	1 GB/s	1.02 millisecond	0	100
1024 MB	1 GB/s	0.004 millisecond	0	100
2048 MB	1 GB/s	0.008 millisecond	0	100
4096 MB	1 GB/s	0.016 millisecond	0	100

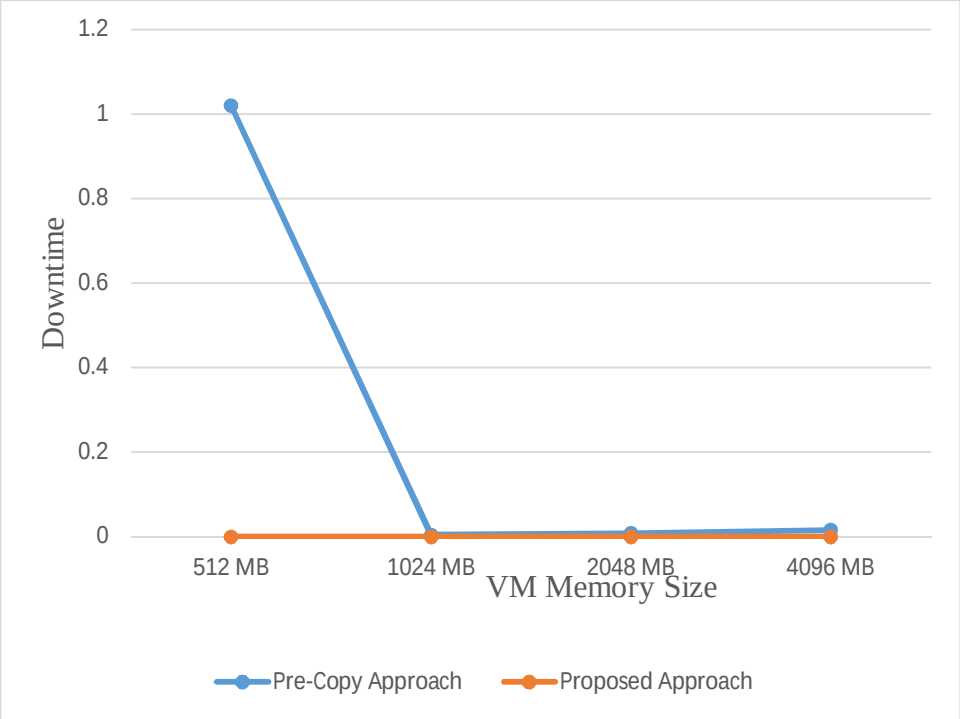


Figure 5-7: -Downtime for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 1 GB Network Bandwidth

Table 5-8: - Downtime for Migrating Different VM Over 100 MB/S Network Bandwidths

VM Memory size	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
512 MB	100 MB/s	2.048 millisecond	0	100
1024 MB	100 MB/s	4.096 millisecond	0	100
2048 MB	100 MB/s	8.19 millisecond	0	100
4096 MB	100 MB/s	16.38 millisecond	0	100

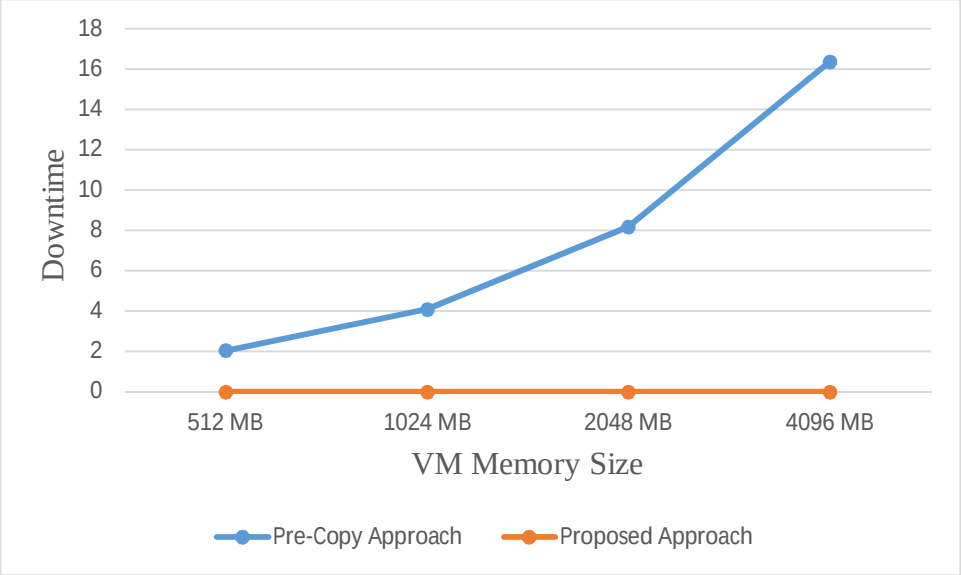


Figure 5-8: - Downtime for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 100 MB Network Bandwidths

Table 5-9: - Downtime for Migrating Different VM Over 1 MB/S Network Bandwidth

VM Memory size	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
512 MB	5 MB/s	167.7 millisecond	0	100
1024 MB	5 MB/s	335.5 millisecond	0	100
2048 MB	5 MB/s	268.4 millisecond	0	100
4096 MB	5 MB/s	214.7 millisecond	0	100

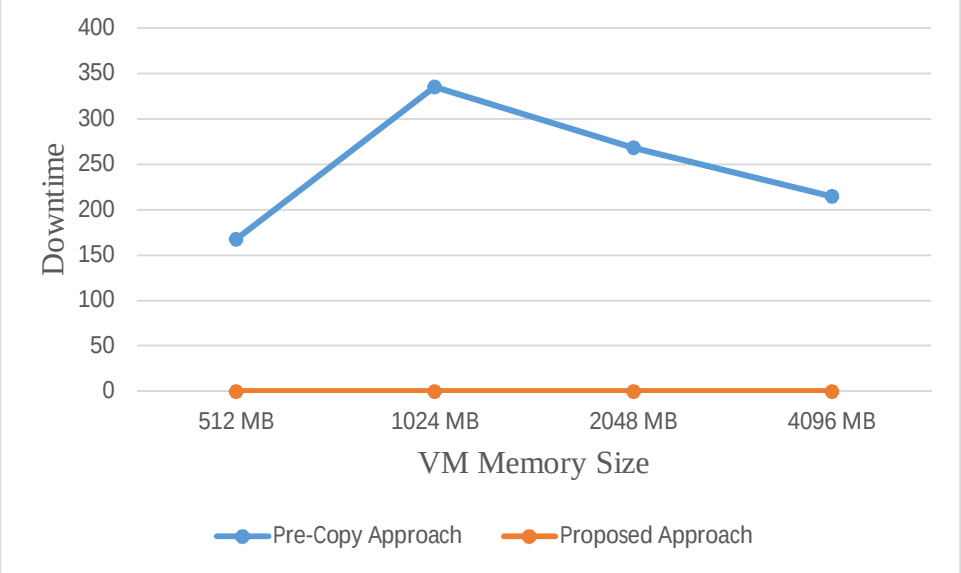


Figure 5-9: - Downtime for Migrating Different VM Using Pre-Copy Approach and Proposed Approach Over 5 MB/S Network Bandwidths

5.2. Evaluation

Live virtual machine migration is migrating a running VM from source host to destination host over a network. To say one VM migration approach is optimal or efficient its total migration time, total page transferred and downtime should be minimum as possible. Nowadays there are several live migration algorithms such as pre-copy approach [3], post-copy approach [4] and MECOM approach [17] etc. but, in this paper, we focused only on pre-copy approach to compare with the proposed approach.

Pre-copy [3] algorithm migrates a running VM from source node to the target node in an iterative fashion. First, it transfers the complete VM to the destination host. In meantime, the VM at source host will update it memory page which is called dirty page. These dirty pages will be transferred in the next round. This iteration will continue until it reaches either a predefined iteration number or a small writable working set has been identified. After this, the stop and copy phase will transfer the remaining dirty pages and VM states to the target host and resume there. Migrating a large size VM over a slow network using the pre-copy approach results a large total migration time, total page transfer and service downtime. To optimize pre-copy approach, we design and implement a new approach. The new proposed algorithm works by migrating both memory and CPU state simultaneously while the VM is executing at the source machine as described in chapter 3 “Design” section 3. 1. The proposed approach migrates a VM with minimum total migration time, total page transfer and downtime related to the pre-copy approach as showed in Table 5-10, Table 5-11 *and* Table 5-12.

Table 5-10: - Total Migration Time for Both Pre-Copy and Proposed Approach

Experiment No.	***VM Memory size***	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
Experiment 1	512 MB	1 GB/s	1.5 second	0.5 second	66.700
	1024 MB		1.03 second	1.02 second	0.970
	2048 MB		2.06 second	2.05 second	0.045
	4096 MB		4.12 second	4.10 second	0.500
Experiment 2	512 MB	100MB/s	7.27 second	5.24 second	27.900
	1024 MB		14.5 second	10.46 second	27.860
	2048 MB		29.08 second	20.9 second	28.129
	4096 MB		58.16 second	41.79 second	28.146
Experiment 3	512 MB	5MB/s	338.15 second	143.8 second	57.470
	1024 MB		676.3 second	287.2 second	57.530
	2048 MB		950.65 second	573.9 second	39.600
	4096 MB		1579.7 second	1147.16 second	27.380

From Table 5-10 we see that the total migration time taken to migrate different size of VM over the different network using both pre-copy approach and proposed approach. This shows the proposed approach reduced total migration time compared to the pre-copy approach. In Table 5-11 the proposed approach reduced the total pages transferred as compared to the pre-copy approach.

Table 5-11: - Total Page Transferred for Both Pre-Copy and Proposed Approach

Experiment No.	***VM Memory size***	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
Experiment 1	512 MB	1 GB/s	131334.1	131334.6	-0.0004
	1024 MB		262669.3	262668.8	0.0002
	2048 MB		525338.6	525337.08	0.0003
	4096 MB		1050677.3	1050673.6	0.0004
Experiment 2	512 MB	100MB/s	133745.86	133698.6	0.0035
	1024 MB		267491.7	267392	0.0037
	2048 MB		534983.47	534778.88	0.0038
	4096 MB		1069966.95	1069552.6	0.0387
Experiment 3	512 MB	5MB/s	218310.16	183603.2	15.0089
	1024 MB		436620.3	367104	15.0092
	2048 MB		873584.26	734105.6	15.0096
	4096 MB		1747443.4	1468108	15.0098

In Table 5-12 our algorithm shows that it eliminates the service downtime by 100% whereas in pre-copy approach downtime increase as the VM size is increased and the network bandwidth decrease.

Table 5-12: - Downtime for Both Pre-Copy and Proposed Approach

Experiment No.	***VM Memory size***	***Bandwidth***	***Pre-copy***	***Proposed approach***	***Reduction ratio (%)***
Experiment 1	512 MB	1 GB/s	1.02 millisecond	0	100
	1024 MB		0.004 millisecond	0	100
	2048 MB		0.008 millisecond	0	100
	4096 MB		0.016 millisecond	0	100
Experiment 2	512 MB	100MB/s	2.048 millisecond	0	100
	1024 MB		4.096 millisecond	0	100
	2048 MB		8.19 millisecond	0	100
	4096 MB		16.38 millisecond	0	100
Experiment 3	512 MB	5MB/s	167.7 millisecond	0	100
	1024 MB		335.5 millisecond	0	100
	2048 MB		268.4 millisecond	0	100
	4096 MB		214.7 millisecond	0	100

Finally, the proposed approach shows better performance than the pre-copy approach. The proposed approach migrates a dissimilar size of VM with minimum total migration time as presented in Table 5-10, total pages transferred as showed in Table 5-11 and zero downtime as displayed in Table 5-12 over different network compared to pre-copy approach. Due to this the proposed approach has improved optimization.

6. CHAPTER SIX: CONCLUSION AND FUTURE WORK

6.1. Conclusion

Cloud computing is a superb computing technology which provides configurable computing resource (e.g. Applications, Services, Servers, Storages, and Network) to cloud users based on the pay-per-use model. Behind the concept of cloud computing, there is a technology which is called Virtualization. Virtualization is a technology that abstracts computing resources (e.g. RAM, CPU and Bandwidth) with the help of hypervisor to run multiple OS simultaneously on a single physical machine.

Live VM migration is one of the virtualization technology capability. Live VM migration is transferring a running VM from one physical machine to another machine. Live migration has been broadly used in load balancing, energy reduction and dynamic resizing to increase availability and hardware maintenance. Basically, during live VM migration performance parameters like total migration time, total page transferred and service downtime is needed to be minimum as much as possible. So far, there are two widely used live virtual machine migration approaches which are called Pre-copy and Post-copy approach. However, existing migration techniques are not silent enough for highly utilized clouds due to their latency and network bandwidth consumption.

In this paper, we design and implement a new live VM migration algorithm which provides a fast and silent migration during highly utilized clouds operation. The proposed approach minimizes the total migration time and total pages transferred, and eliminate the service downtime as compared to the existing approach. To proof this, we created a cloud computing environment using CloudSim [25] framework and implemented both pre-copy approach and the proposed approach in java programming language. We conducted totally 24 simulations (12 simulations for each approach) which are grouped into three different type of experiments.

The performance result of both pre-copy approach and proposed approach analyzed and presented visually using experiments snapshot, complex graphs, and tables. Based on the result we get from the experiment the proposed approach has better efficiency and performance as compared to the pre-copy approach. Also, the proposed approach answered all specific objective and research question stated at the beginning of this study.

Finally, during this study, we have learned several cloud architectures, live VM migration approaches, and their implementations. This paper also gave us a very useful experience in working with cloud computing simulation tools like CloudSim [25] which give us the ability to create and manage cloud computing environment easily. We also explore different hypervisor like KVM and Xen to understand how live VM migration works in a real environment using the existing system such as pre-copy and post copy approach. Because of cloud computing is today's hot topic for the future IT solutions this study will demonstrate to be a very good experience and help us drive our own future career towards cloud technology.

6.2. Future Work

In this work, a new VM migration algorithm is developed and implemented to migrate a VM between hosts while the VM is running. Due to time and scope limitations, the proposed approach migrates only one running VM from source host to the target host within LAN network. However, to produce a more realistic result the following points are needed to be addressed in the future but, other students and researchers also can explore one or more of these topics.

- In the future, we will implement and deploy the proposed algorithm in a real environment. To get the accurate result that much the simulation result.

- This study deals only with migrating a single running virtual machine between two hosts. However, for the future designing and implementing a new algorithm that enables live migration of multiple running virtual machines with minimum total migration time and zero downtime is crucial in cloud computing environment. Because of this cloud providers can reduce costs and deliver quality services to their cloud customers.

- The new proposed algorithm migrates a running virtual machine from source host to the target host within local area networks only. For the future, we will extend our research by improving the proposed algorithm to migrate a virtual machine within a wide area network.

- All currently available algorithms and the proposed algorithm migrates a virtual machine that runs a non-real-time application. In the future, designing and implementing an algorithm that migrates a virtual machine while it executes a real-time application can be a good research area.

REFERENCE

[1] P. Mell and T. Grance, "The NIST Definition of Cloud," National Institute of Standards and Technology, 2011.

[2] C. Rajkumar B., "Morgan Kaufmann is an imprint of Elsevier, ,," in *Mastering Cloud Computing Foundations and Applications Programming*, 225 Wyman Street, Waltham, MA 02451, USA, Elsevier Inc, 2013.

[3] K. F. S. H. J. G. H. E. J. C. L. I. P. A. W. Christopher Clark, "Live Migration of Virtual Machines," p. 14, 2005.

[4] U. D. a. K. G. Michael R. Hines, "Post-Copy Live Migration of Virtual Machines," p. 13, 2009.

[5] J. McCarthy, "REMINISCENCES ON THE H ISTORYOF," Stanford University, 1983.

[6] G. Reese, Cloud Application Architectures, 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2009.

[7] B. Sosinsky, Cloud Computing Bible, 10475 Crosspoint Boulevard ,Indianapolis, IN 46256: Wiley Publishing, Inc., 2011.

[8] B. Furht and A. Escalante, Handbook of Cloud Computing, Florida Atlantic University: Springer, 2010.

[9] L, Wag, G. V. Laszewski, A. Younge , X.He , M. Kunze , J. Tao , C. Fu,, "Cloud Computing: a Perspective Study," *Journal of New Generation Computing*, vol. 28, p. 11, April 2010.

[10] "http://www.cloudcontrols.org/wp-content/uploads/2011/06/NIST_Visual_Model_of_Cloud_Computing_Definition.jpg," [Online].

[11] "http://www.whatiscloud.com/cloud_deployment_models/private_clouds," [Online].

[12] "http://whatiscloud.com/cloud_deployment_models/community_clouds," [Online].

[13] "http://www.whatiscloud.com/cloud_deployment_models/public_clouds," [Online].

[14] "http://whatiscloud.com/cloud_deployment_models/hybrid_clouds," [Online].

[15] "SUSE Linux Enterprise Server 12 SP2: Virtualization Guide," 2016. [Online]. Available: https://www.suse.com/documentation/sles-12/singlehtml/book_virt/book_virt.html#sec.virtualization.introduction.benefits.

[16] W. G. S. W. X. S. X. W. a. F. Z. Hai Jin, "Optimizing the live migration of virtual machine by CPU scheduling," p. 9, 9 July 2010.

[17] L. S. X. H. a. X. HaiJin, "MECOM: Livemigration of virtual machines by adaptively compressing memory pages," p. 13, 23 October 2013.

[18] E. P. Z. a. N. L. Thein, "Live Virtual Machine Migration with Efficient Working Set Prediction," vol. vol.11, p. 6, 2011.

[19] A. M. a. S. S, "An Optimized Approach for Live VM Migration using Log Records," p. 4, 2013.

[20] S. S. a. V. Varma, "A Hybrid Approach To Live Migration Of Virtual Machines," p. 5, 2012.

[21] J. C. F. R. M. E. S. Q.-O. ´. Francisco J. Clemente-Castell´o, "Adaptive Downtime for Live Migration of Virtual Machines," p. 8, 2014.

[22] M. R. D. a. H. B. Patel, "Efficient Virtual Machine Migration in Cloud Computing," p. 5, 2015.

[23] P. B. G. G. G. ABRAHAM SILBERSCHATZ, "," in *Operating Systems Concepts* , 2013, p. 944.

[24] A. S. Tanenbaum, "Modern Operating Systems,," 2008.

[25] R. R. A. B. C. A. F. D. R. a. R. B. Rodrigo N. Calheiros, "CloudSim: a toolkit for modeling and

simulation of cloud computing environments and evaluation of resource provisioning algorithms," p. 28, 2010.

[26] P. B. Y. A. a. S. U. K. Dzmitry Kliazovich, "GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers," p. 5, 2010.

[27] S. K. G. a. R. Buyya, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations," p. 9, 2011.

[28] R. N. C. a. R. B. Bhathiya Wickremasinghe, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications," p. 7, 2010.

[29] R. N. C. a. D. G. G. Thiago Teixeira Sá, "Environments, CloudReports: An Extensible Simulation Tool for Energy-Aware Cloud Computing," p. 16, 2014.

[30] M. J. M. k. Z. A. M. N. A. a. M. A.-A. Yaser Jararweh, "CloudExp: A comprehensive cloud computing experimental framework," p. 13, 2014.

[31] A. N. ·. J. L. V.-P. ·. A. C. C. ·. G. G. C. ·. J. C. ·. I. M. Llorente, "iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator," p. 26, 2012.

[32] M. A. S. N. C. A. F. D. R. a. R. B. Rodrigo N. Calheiros, "EMUSIM: An Integrated Emulation and Simulation Environment for Modeling, Evaluation, and Validation of Performance of Cloud Computing Applications," p. 18, 2012.

APPENDICES

Appendices A: Sample Source Code

Create Datacenter: -

```
public static Configuration createDatacenter(String name) {

    //   Create a DatacenterCharacteristics object that stores the

    //    properties of a data center: architecture, OS, list of

    //    Machines, allocation policy: time- or space-shared, time zone

    //    and its price (G$/Pe time unit)

    String arch = "x86";      // system architecture

    String os = "Linux";          // operating system

    String vmm = "Xen";

    double time_zone = 10.0;         // time zone this resource located

    double cost = 3.0;              // the cost of using processing in this resource

    double costPerMem = 0.05;                    // the cost of using memory in this resource

    double costPerStorage = 0.001;       // the cost of using storage in this resource

    double costPerBw = 0.0;                      // the cost of using bw in this resource

    LinkedList<Storage> storageList = new LinkedList<Storage>();      //we are not adding SAN devices by now

    DatacenterCharacteristics characteristics = new DatacenterCharacteristics(

          arch, os, vmm, createHost(), time_zone, cost, costPerMem, costPerStorage, costPerBw);

    // Finally, we need to create a PowerDatacenter object.

    Configuration datacenter = null;

    try {

       datacenter = new Configuration(name, characteristics, new VmAllocationPolicySimple(createHost()), storageList, 0);

    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return datacenter;
    }
```

Create Broker: -

```
public static NewBroker createBroker() {
NewBroker broker = null;
try {
broker = new NewBroker("Broker");
} catch (Exception e) {
e.printStackTrace();
return null;
}
return broker;
}
```

Create Host: -

```
public static List<Host> createHost() {
//      Create Host with its id and list of PEs and add them to the list of machines
        hostList = new ArrayList<Host>();
        int ram = 8192; //host memory (MB)
        long storage = 1000000; //host storage(1tera)
        int bw = 10000;
        List<Pe> peList = createPeList();
        for (int i = 0; i < 2; i++) {
            int hostId = i;
```

```
            hostList.add(
                    new Host(
                            hostId,
                            new RamProvisionerSimple(ram),
                            new BwProvisionerSimple(bw),
                            storage,
                            peList,
                            new VmSchedulerTimeShared(peList)
                    )
            );
            //peList = createPeList().subList(4, 5);
        }
        return hostList;
    }
```

Create Virtual Machine: -

```
public static List<Vm> createVM() {
        vmlist = new ArrayList<Vm>();
        //Create one virtual machine
        //VM description
        int mips = 500;
        long size = 10000; //image size (MB)
        int ram = 1024; //vm memory (MB)
        long bw = 1000;
        int pesNumber = 1; //number of cpus
        String vmm = "Xen"; //VMM name
```

```
        for (int i = 0; i < 2; i++) {
            int vmid = i;
            //create two VMs
            //add the VMs to the vmList
            vmlist.add(new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared()));
            //mips = mips * 2;
        }
        return vmlist;
    }
```

Create CloudLet: -

```
public static List<Pe> createPeList() {
        List<Pe> peList = new ArrayList<Pe>();
        int mips = 1860;
        for (int i = 0; i < 2; i++) {
            // Create PEs and add these into a list.
            peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating
        }
        return peList;
    }
```

VM Migration Map: -

```
public static Map<String, Object> getMigrationMap() {
    Map<String, Object> migrationMap = new HashMap<String, Object>();;
    Vm vm = VmList.getById(vmlist, 0);
    Host targetHost = HostList.getById(hostList, 1);
    Host sourceHost = (Host) vm.getHost();
    migrationMap.put("vm", vm);
    migrationMap.put("host", targetHost);
    if (vm.getId() == targetHost.getId()) {
       migrationMap.clear();
       Log.printLine("VM allocation to the destination host failed");
       System.exit(0);
    } else {
       targetHost.addMigratingInVm(vm);
    }
    return migrationMap;
```