VM Selection Process Management for Live Migration in Cloud Data Centers

Suhib Bani Melhem

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

December 2017

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Suhib Bani Melhem**

Entitled: **VM Selection Process Management for Live Migration in Cloud Data Centers**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. A. Awasthi

_____ External Examiner
Dr. F. Gagnon

_____ External to Program
Dr. R. Dssouli

_____ Examiner
Dr. Y. Liu

_____ Examiner
Dr. D. Qui

_____ Thesis Supervisor
Dr. A. Agarwal

Approved by_____
Dr. W.E. Lynch, Chair of Department

February 6, 2018 _____
Dr. A. Asif, Dean of Faculty of Engineering and Computer Science

ABSTRACT

VM Selection Process Management for Live Migration in Cloud Data Centers

Suhib Bani Melhem, Ph.D.

Concordia University, 2017

With immense success and fast growth within the past few years, cloud computing has been established as the dominant computing paradigm in information technology (IT) industry, wherein it utilizes dissipated resource benefits and supports resource sharing and time access flexibility. The proliferation of cloud computing has resulted in the establishment of large-scale data centers across the world, consisting of hundreds of thousands, even millions of servers. The emerging cloud computing paradigm provides administrators and IT organizations with considerable freedom to dynamically migrate virtualized computing services among physical servers in cloud data centers.

Normally, these data centers incur very high investment and operating costs for the computing and network devices as well as for the energy consumption. Virtualization and virtual machine (VM) migration offers significant benefits such as load balancing, server consolidation, online maintenance and proactive fault tolerance along data centers. VM migration relies on how to determine the trigger condition of VM migration, select the target virtual machine, and choose the destination node.

As a result, dynamic VM migration in the scope of resource management is becoming a crucial issue to emphasize on optimal resource utilization, maximum

throughput, minimum response time, enhancing scalability, avoiding over-provisioning of resources and prevention of overload to make cloud computing successful. Intelligent host underload/overload detection, VM selection, and VM placement are the primary means to address VM migration issue. Therefore, these three problems are considered to be the most common tasks in VM migration.

This thesis presents novel techniques, models, and algorithms, for distributed dynamic consolidation of virtual machines in cloud data centers. The goal is to improve the utilization of computing resources and reduce energy consumption under workload independent quality of service constraints. The proposed approaches are distributed and efficient in managing the energy-performance trade-off.

ACKNOWLEDGEMENTS

To a great degree, all praises to god for giving me the power to accomplish this thesis. I could not finish this work without its blessing and guidance. I would like to express my heartfelt gratitude to my supervisor Prof. Anjali Agarwal who presided over this thesis and in that line improved it significantly. I have to express out appreciation to the committee members for evaluating my thesis. Many thanks to my close friend, Mufleh Al-Shatnawi, for stimulating discussions and good memories shared together. Also, I would like to thank my friend Mustafa Daraghmeh. And I would like to thank Cistech Limited for their financial support. Last but not the least, I would like to express my profound gratitude to my beloved family, my wife, parents, sisters, brothers, who expressed their encouragement and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AC	: Available Capacity
ACO	: Ant Colony Optimization
BST	: Binary Search Tree
DFQL	: Dynamic Fuzzy Q-Learning
FCM	: Fuzzy C-Mean
FGA	: Family Genetic Algorithm
FOMCHSD	: First Order-Markov Chain Host State Detection algorithm
HFD	: Host Fault Detection
HGA	: Hybrid Genetic Algorithm
HPG	: Highest Potential Growth
IaaS	: Infrastructure as a Service
IQR	: InterQuartile Range
LAN	: Local Area Network
LiRCUP	: Linear Regression-based CPU
LR	: Local Regression
LRR	: Local Robust Regression
MAD	: Median Absolute Deviation
MadMCHD	: Median Absolute Deviation Markov Chain host Detection algorithm
MC	: Maximum Correlation
MDBP	: Multidimensional Bin-Packing
MDL	: Migration Delay
MedianMT	: Median Migration Time
MiMc	: Minimum VM Migrated Count
MIPRT	: Minimum IP Reconfiguration Time
MIPS	: Millions Instructions Per Second
MM	: Minimization of Migration
MMT	: Minimum Migration Time
MmtMiMc	: Minimum Migration Time Minimum VM Migrated Count
MMTMU	: Minimum Migration Time Maximum CPU Utilization algorithm
MMTMUR	: Minimum Migration Time Maximum User ratio
MPABFD	: Markov Power Aware Best Fit Decreasing
MU	: Maximum Utilization
MUDC	: Data Center Minimum Utilization
MUIPRT	: Minimum Utilization minimum IP Reconfiguration Time
OS	: Operating System
PaaS	: Platform as a Service
PABFD	: power Aware Best Fit Decreasing
PDM	: Performance Degradation due to Migration
RS	: Random Selection
RSDC	: Data Center Random Selection
SaaS	: Software as a Service
SLA	: Service Level Agreement

SLATAH	: SLA violation Time per Active Host
VMCUP	: TOPSIS-Available Capacity-Number of VMs-migration Delay
VMM	: Threshold-based algorithm
VPLS	: Utilization and Minimum Correlation
VPNs	: Virtual Computing lab
WAN	: VM-based Dynamic Threshold

Chapter 1

Introduction

Cloud computing is a fast-growing computing technology. It is defined by NIST [1] organization as a "Model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (network devices, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Several other definitions have been proposed for cloud computing, but they all imply the existence of a shared pool of computing resources. In cloud computing, physical servers are referred to hosts, whereas the group of hosts and storages connected by network devices is referred to a data center, and each host contains several numbers of virtual machines. Virtual machine (VM) represents an entire operating system (OS) with its associated applications and services.

In cloud computing, the applications and services are accessible to clients over the internet remotely. As shown in Figure 1-1, the services provided by cloud computing can be classified as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These services are offered available as pay-as-you-go model to clients. The most popular examples are Google's App Engine [2], Amazon's

EC2 [3], Microsoft Azure [4], and IBM SmartCloud [5]. As shown in the figure, the cloud computing can be deployed as a private cloud, a public cloud, or as a hybrid cloud.

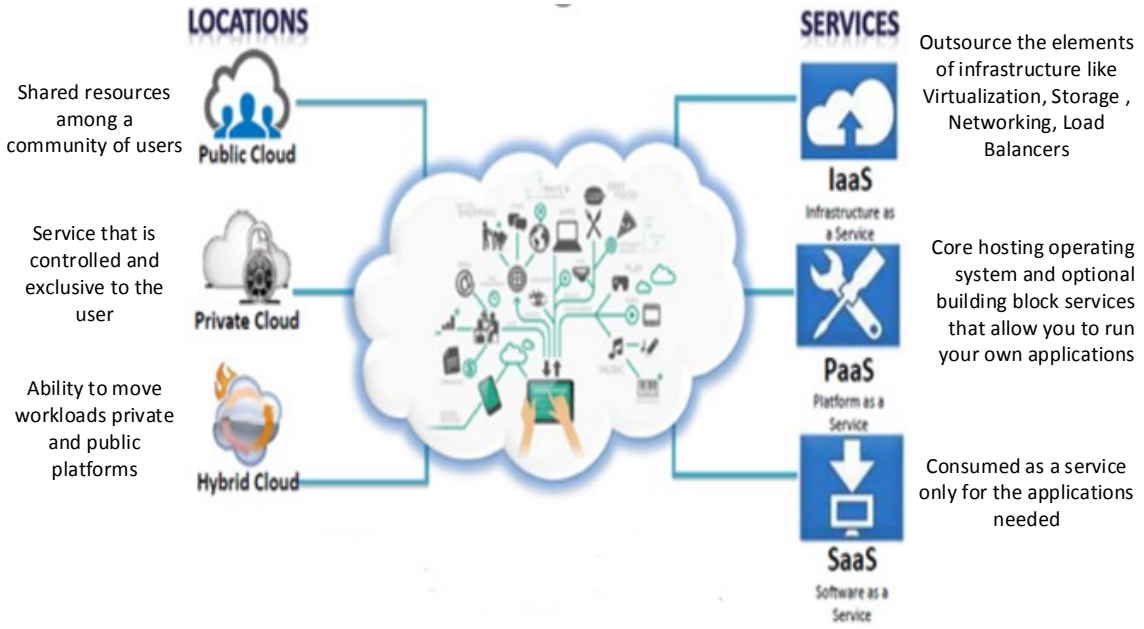

Figure 1-1: Cloud Computing [6]

Cloud computing is based on the concept of virtualization. Virtualization plays a vital role in managing and organizing access to the resource pool via a software layer called virtual machine monitor (VMM) or hypervisor. It hides the details of the physical resources and provides virtualized resources for high-level applications. Besides, it virtualizes all of the resources of a given host allowing several VMs to share its resources [7]. VMware ESX/ESXi [8], Virtual PC [9], Xen [10], Microsoft Hyper-V [11], KVM [12], and VirtualBox [13] are popular virtualization software. Virtualization also allows gathering several VMs into a single host using a technique called VM consolidation. Another capability provided by virtualization is live migration, which is the ability to transfer a VM between hosts.

The scope of this work focuses on a IaaS module, which handles infrastructure resources (virtual machines, servers, storage, and network) allocation, provisioning, requirement mapping, adaptation, discovery, brokering, estimation, and modeling. Resource management for IaaS in cloud computing offers following benefits: scalability, quality of service, optimal utility, reduced overheads, improved throughput, reduced latency, specialized environment, cost effectiveness and simplified interface. With the rise of cloud computing, a huge complexity growth of the structure happens. Therefore, to effectively manage applications and resources it is crucial to use the models and tools that create an application profile, which is used to apply optimal models to determine the most suitable amount of resource for each workload. Virtual machines migration is one of the most popular ways to manage resources, and live VM migration is the most used technique to reload or rearrange the resources in the data center to keep the delivered services available. Live VM migration is defined as a technique that migrates the entire OS and its associated applications from one host to another where a user does not notice any interruption in his service. Live VM migration plays an important role to facilitate online maintenance, load balancing, and energy management as part of resource management [14].

What is a Cloud Data Center?

The data centers consist of network equipment like routers, switches, cabinets, servers, and electrical equipment like switchgear, PDUs, UPS, CRAC, generators and HVAC systems [15]. Typically, conventional data centers are provisioned to satisfy the peak demand, which results in wastage of resources during non-peak periods. Modern-day data

centers are turning to the cloud-based to mitigate the above problem. The essential characteristics of cloud-based data centers are [16]:

- Making resources available on demand. The operation and maintenance of the data center lie with the cloud provider. Thus, the cloud model enables the users to have a computing environment without spending an enormous amount of money in building a computing infrastructure.
- Flexible resource provisioning. It provides the ability to scale dynamically or shrink the provisioned resources as per the dynamic requirements.
- Fine-grained metering. It enables the "pay-as-use" model, so the users do not need to stay into long-term contracts since users pay only for the services used.

However, implementing cloud-based data centers demands an enormous deal of flexibility and agility for both the users and providers. For instance, the dynamic scaling and shrinking requirement needs compute resources to be made available at a very short notice.

IaaS Cloud System

One of the various definitions of "cloud" is that of an Infrastructure-as-a-Service (IaaS) system, which enables on-demand provisioning of computational resources via VMs deployed in a cloud provider's data center [17]. It was first popularized in 2006 by Amazon's Elastic Compute Cloud (EC2) [3], which started to offer virtual machines (VMs) for US$0.10 an hour using both a simple Web interface and a programmer-friendly API. Amazon EC2 contributed to popularizing the IaaS paradigm, although not the first to propose a utility computing model, it became closely tied to the notion of cloud computing [18].

Virtual Computing Lab (an IaaS Cloud Module)

Virtual Computing Lab (VCL) is a free, cloud computing project, open-source, on-demand, remote-access system that dynamically provisions computing resources to end users [19, 20]. North Carolina State University in cooperation with IBM announced the creation of the system in 2006 with the goal of creating a multi-institutional, shared computing services community, which includes universities, colleges, schools and business partners. It became an Apache Project in 2008 [21], and then an Apache Software Foundation top-level project in 2012 [20], which provides an open cloud environment for educational purposes.

VCL has high throughput architecture of computational power, which can keep track of all its computation nodes and redistribute the VM from a heavy loaded node to the least utilized physical computation node. The VCL framework has been chosen to deploy an educational cloud environment where availability of the resources anywhere and anytime is the most significant advantage of the VCL solution. Moreover, many other benefits can be summarized as follows:

- Raising computing resource accessibility.
- Increasing integrity and availability of data, applications, and research materials.
- Increasing end users' mobility to make resources accessible anywhere and anytime.
- Reducing client application and the system resource footprint.
- Increasing application and computing performance utilization.
- Providing convenient web access and a self-service portal.

VCL is considered one of the open-source Cloud Computing solutions, which differ by different criteria like architecture, functionality, purpose of use, and target clients served. Table 1-1 compares eight kind of IaaS open-source clouds in an abstract way concerning hypervisor used (infrastructure) attribute, and the main characteristic attribute. VCL aims to develop and promote virtualization concepts and open-source solutions for the benefit of the academia and its stakeholders – by creating shared virtual computing resources and supporting related research. VCL has been deployed, from an academic perspective, to provide services to students and academic staff as well [22].

Table 1-1: Comparison between IaaS open-source clouds Computing Solutions

Solutions	***Infrastructure***	***Characteristic***
XCP[23,24]	Xen	Aims to turn legacy clusters into IaaS Clouds
Nimbus[23,24]	Xen, KVM	Only a tool for automatic maintenance of cloud
OpenNebula[23,24]	Xen , KVM , VMware	Grouping nodes to allow HPCaaS
Eucalyptus[23]	Xen, KVM	Hierarchical Architecture
VCL[21]	VMware, Xen, KVM	Offer capabilities that are very flexible and diverse
Enomaly[23,24]	Xen, VirtualBox, KVM, VMware	Open version is focused on small clouds
OpenStack[25]	Xen,KVM,VMware,Hyper-V	Modular platform, complete solution for cloud computing
CloudStack[26]	Xen, VMware,KVM	Complete solution for cloud computing

Live VM Migration

There is a need to reorganize the VMs and the hosts to provide load balancing or server consolidation depending on the service level agreement (SLA) with the end users and other issues. Live virtual machines migration is one of the most popular ways to manage

resources to keep the delivered services available. The benefits of VM migration include server consolidation, load balancing among the physical servers (hosts) and failure tolerance in case of sudden failure.

Live VM migration is divided into two parts: 1) Selection process which involves when to start the migration process, determining which VM must be selected to be migrated, which destination host must be chosen to move this VM. The goals of the selection process are to reduce power consumption, load balancing, and improve fault tolerance, which eventually will increase the cloud productivity, services availability and throughput, and reduce its operation cost, pollution (green data centers), and hardware maintenance. 2) Migration Process that targets moving the VM in minimum time to avoid any interruption of services. The process can be divided into two categories: a) Suspend /Resume migration, which is used mainly for VM migration through wide area network (WAN), and b) Pre-copy and Post-copy methods used for local area network (LAN) VM migration.

The first phase of live VM migration is the selection process phase which consists of three phases. In the first phase, host detection, a host may be in an overloaded state or in an underloaded state. If a host is underutilized, then all the VMs from this host can be migrated and the host will go to sleep/shutdown mode, or the host will be considered as a good candidate to receive the migrated VMs from the overloaded hosts in the future. On the other hand, when a given host is overloaded some VMs must be selected to migrate from this host to other hosts. The challenges in the host overload/underload detection are to reduce the power consumption, minimize SLA violation, and to avoid performance degradation.

Once a decision to migrate VMs from a given host is made, the second phase, VM selection phase, selects one or more VMs from the full set of VMs running on the host. The selected VMs must be moved to other hosts. VM selection approaches are different based on the parameters that are considered to select the migrated VMs. The challenge in choosing one or more VMs for migration is a vital decision for resource management. The VM migration process consumes network bandwidth and CPU resources from both source and destination hosts besides making the VM unavailable for a certain amount of time. The performance of other VMs that are running on source and destination hosts are also affected due to increased resource demands during the VM migration process.

Finally, in the third phase a given VM placement algorithm is applied to selected underloaded hosts to receive the migrated VMs. Many factors should be considered to develop a new optimal VM placement algorithm, such as the resource availability of host (i.e., CPU, memory, disk storages and network bandwidth), the total energy consumption in the data center, and inter-VM traffic. The goal of VM placement is to deliver best possible QoS to the applications running on VMs. Once a decision to migrate a VM from a source host to a destination host is known as a result of the selection process, then the migration process will take place either locally or widespread.

Algorithm 1-1 illustrates the overall live migration procedure based on the host status which can be either an overloaded or an underloaded state. In the overload host detection procedure, one of the host detection algorithms is applied to determine if the host is overloaded. In case a host is overloaded, a Boolean variable called $migration\_decision\_overloaded$ is continuously checked until the required number of VMs are selected and stored in $vmstomigrate[]$ array. The selection can be

done using any VM selection algorithm. Active hosts that currently carry VMs are determined using $getactivehosts$ function. One of the VM placement algorithms is applied to map selected VMs to destination hosts.

In the underload host detection procedure, one of the host detection algorithms is applied to determine if the host is underloaded. In case the host is underloaded, there is no need of VM selection phase because all the VMs in the underloaded host must be migrated. All these VMs are then mapped to suitable destination hosts based on a placement algorithm. The underloaded host is switched to an idle state.

Algorithm 1-1: Live Migration Procedure.

```
1   Input: Host
2   Output: Do certain procedure based on the host status

3   //Overloaded host detection procedure
4   boolean migration_decision_overloaded ← false
5   boolean selection_is_done ← false
6   migration_decision_overloaded
                ← DetectionPolicy(host_is_overloaded())
7   while migration_decision_overloaded = true do
8       //One of the VM selection algorithms is applied
        vmstomigrate[] ← SelectionPolicy(host).add
9       migration_decision_overloaded ←
                DetectionPolicy(host_is_overloaded())
10      selection_is_done ← true

11  if selection_is_done = true then
12      Activehosts[] ← getactivehosts()
13      PlacementPolicy(vmstomigrate[], Activehosts[])

14  //Underloaded host detection procedure
15  boolean migration decicion_underloaded ← false,
16  migration_decision_underloaded
                ← DetectionPolicy(host_is_overloaded())
17  if migration decision_underloaded = true do
18      vmstomigrate[] ← all vms
19      Activehosts[] ← getactivehosts()
20      PlacementPolicy(vmstomigrate[], Activehosts[])
```

The second phase of live VM migration is the migration process. Figure 1-2 shows the migration process that migrates the entire operating system and its associated

applications from one host to another that may take place either locally or widespread. VM migration over WAN differs from LAN. Firstly, LAN migration leads to transfer of memory state only, whereas WAN transfers the state of local disks as well [27]. Secondly, network reconfiguration is an issue in the WAN migration, migrating into another subnet obliges the server to get a new IP address and, subsequently, it disconnects existing network connections. In this case, the network addresses must be maintained, or the network reconfiguration is implemented by the applications.

The performance metrics that are generally considered to measure the performance of live migration are [28]: 1) total migration time which represents the total time required to move the VM between hosts, 2) downtime which represents the portion of total migration time when the VM is not running, that is the time between pausing the VM on the source and resuming it on the destination, 3) application degradation which represents the extent to which migration slows down the applications executing within the VM.

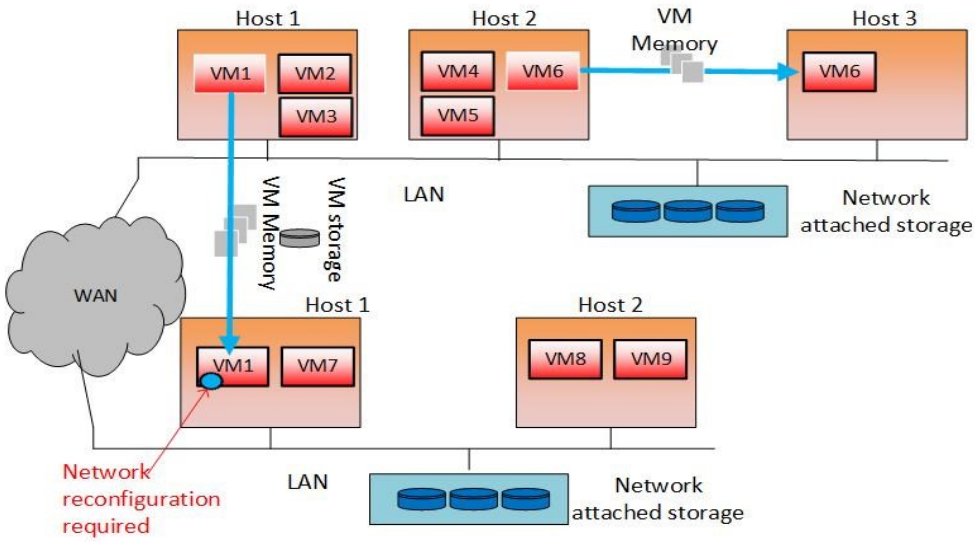

Figure 1-2: VMs migration over LAN/WAN

1.1 Motivation

In modern data centers, resource management and allocation during VM migration is getting more challenging every day due to their rapid growth, high dynamics of hosted services, resource elasticity, and guaranteed availability and reliability. Thus, the performance of applications in large virtualized data centers is highly dependent on data center architecture and smooth network communication among VMs, while minimizing the communication burden to avoid congestion, latency, etc. The communication cost of a network can be reduced by minimizing the VMs migration between hosts. Therefore, clients and service providers need to build a cloud computing infrastructure that does minimize not only operational costs but also total network load. It should be noted that the key aspect that is directly related to network resource management in the data center is how to minimize network overhead, and load balancing-VM migration, which is still an active area of research.

The resource utilization of a data center may change over time due to a creation of new VMs and/or hosts, or due to a failure of existing hosts, or due to the removal of existing VMs. There is a need to reorganize the VMs and the hosts to provide load balancing or server consolidation depending on the SLA with the end users and other issues. In cloud data center management, the three most important research problems that have been addressed are the host underload/overload detection, VM selection, and VM placement.

Host overload/underload detection and VM Placement: There are three main reasons behind our motivation for the first and third problem. First, none of the existing techniques consider a dynamic utilization threshold and predict the future CPU utilization

simultaneously, Second, none of the existing algorithms considered the future expectation of CPU utilization to be underloaded or normal loaded. Third, none of the existing algorithms considered the future expectation of CPU utilization to be underloaded or normal loaded. In contrast, our proposed solution uses historical data to build probabilistic model that can predict the future host load more efficiently. We present a Markov-based VM placement and host load detection approaches, respectively with the objective to allocate a VM based on the current and future resource utilization of host and VMs to mitigate the unneeded VM migrations for better SLA violation, number of VM migrations and the energy consumption in the whole system.

VM Selection: The main reason behind our motivation for this problem is that none of the existing algorithms consider the number of migrations related to the VM. The existing VM selection algorithms focused on minimizing SLA violation on all the system and they ignore the frequent violation for the same VM, where a certain VM might be selected frequently to migrate from its overloaded host to another host based on the VM selection policies.

Proactive selection process in live VM migration across cloud data centers:

In cloud data center management, many techniques have been proposed over the last years to solve selection process research problem, but these techniques are more restricted for LAN live VM migration. In a sense, they are assuming that there is no need for IP reconfiguration during the live VM migration. None of the existing algorithms take in its consideration the number of users currently connected to a given active host. None of the existing algorithms consider which data center must be chosen as a target to

receive the migrated VMs from the overloaded data centers, and no proactive criteria exists for live WAN migration that minimizes IP reconfiguration time.

It should be noted that it is necessary to migrate the VMs to different data centers that are located at different geographic locations (i.e. different subnet configurations) to obtain high QoS. Thus, WAN live VM migration techniques have been proposed [16, 29-38]. It is known that when a given VM moves to a new subnet (i.e., new data center existing in a different LAN), a mobility solution or scheme should be applied to maintain the network connectivity and to preserve the open connections during and after the migration. This migration forces the VM to get a new IP address, and as a result breaks existing network connection. Therefore, WAN live VM migration results into a mobility problem, which may render the service unreachable and increase the downtime of VM during the migration process. There are a few techniques proposed to solve IP network reconfiguration [33, 36-39], but the existing techniques focused on applying a mobility solution or scheme to maintain the network connectivity and to preserve the open connections during and after the migration in the migration process.

1.2 Research Problems and Objectives

This thesis tackles the research challenges that are not only related to LAN VM selection process but also are related to WAN VM selection process. The VM selection process deals with the following subproblems: determine which hosts or data centers are overloaded (i.e., when to migrate), determine which VMs must be selected to be migrated, and determine which hosts or data centers must be chosen to receive the migrated VMs (i.e., where to migrate).

In the following subsections, the subproblems of the VM selection process will be discussed.

- **Host Detection.** The host detection can be divided into overload host detection and underload host detection. If a host is underutilized, then all the VMs from this host can be migrated and the host will go to sleep/shutdown mode to improve the utilization of resources or the host will be considered as a good candidate to receive the migrated VMs from the overloaded hosts in the future. On the other hand, the host overload is the process of determining when a given host is overloaded so that some VMs must be selected to migrate from this host to other hosts to avoid performance degradation. A crucial decision that must be made in both situations is determining the best time to migrate VMs to minimize energy consumption, while satisfying the defined QoS constraints.
- **LAN/WAN migration:** It is the decsion to make a migration in the same data center, which is called LAN migration, or in a different data center, which is called WAN migration. WAN migration is useful in many cases even though it has overhead related to network reconfiguration process and costs associated with storage migration. One of the reasons for WAN migration is when a data center is considered to be overloaded and one or more VM migration is required from data center under consideration.
- **VM Selection.** Once a decision to migrate VMs from a given host is made, a particular VM selection algorithm should be applied to select one or more VMs from the full set of VMs running on the host. The problem consists of determining

the best subset of VMs to migrate that will provide the most beneficial system reconfiguration.

- **Data Center Selection.** Once the decision to migrate the VM from a given data center is made, it is necessary to find the most suitable data center to receive the migrated VM.
- **VM Placement.** A given VM placement algorithm is applied to select underloaded hosts or to receive the migrated VMs from the overloaded hosts or data centers. The VMs are migrated to another host when the current host cannot meet the resource requirements. Determining the best placement of new VMs or the VMs selected for migration to other servers is another essential aspect that influences the quality of VM consolidation and energy consumption by the system.

To deal with the challenges associated with the above research problems, the following objectives will be delineated:

- Explore, analyze, and classify the research in the area of resource data center management to gain a systematic understanding of the existing techniques and approaches.
- Conduct competitive analysis of selection process algorithms to insights into the design of algorithms for dynamic VM consolidation and load balancing-VM migration. This analysis aims to determine the factors that lead to an optimal resource utilization, maximum throughput, maximum response time and prevention of overload situation.
- Propose an algorithm for dynamic host overload/underload detection. The proposed algorithm considers metrics such as hosts threshold, SLA violation, performance degradation, and the number of VMs migration.
- Propose algorithms for dynamic LAN and WAN VM selection. The proposed algorithm considers metrics such as SLA violation, performance degradation, the

number of VMs migrating, and in case of WAN migration the needed number of IP reconfiguration.

- Propose an approach to design a dynamic LAN and WAN VM placement system in a distributed manner. The proposed algorithm will consider metrics such as SLA violation, performance degradation, number of migration, and service downtime.
- Compare the results of proposed algorithms with those of the other algorithms in the literature using CloudSim simulator.

1.3 Contributions

The contributions of this thesis can be generally divided into 4 categories: classification and analysis of the area, novel model and algorithms for host load detection and VM placement in Live Migration, novel algorithms for minimizing biased VM selection in live VM migration, and design and implementation of a a system model to provide proactive selection process for live VM migration across cloud data centers. The key contributions are:

1.3.1 A taxonomy and survey of the state-of-the-art approaches used in the live VM migration (Chapter 2)

- Compares existing host detection algorithms.
- Compares existing VM selection algorithms.
- Compares existing VM placement algorithms.
- Present tchniques in solving IP address change due to WAN migration process.

1.3.2 Prediction Model for Host Detection and VM Placement (Chapter 3)

- In contrast to the existing VM consolidation and load balancing methods which mostly rely on the current resource utilization of hosts, Markov chain model considers both current and future resource utilization. In order to predict the future utilization, we used the first-order Markov chain model to build Markov host prediction model.
- Propose a host load detection algorithm called Median Absolute Deviation Markov Chain Host Detection algorithm (MadMCHD) to find the future overutilized hosts state and the future underutilized hosts state for better host detection performance in the live migration. In addition, we propose an efficient prediction algorithm to enhance VM placement process. We improve the live migration process by combining the proposed algorithms for better performance.
- Implement and evaluate Markov host prediction model and the algorithms on a simulated large-scale data center using the real PlanetLab workloads and a random workload.
- Study the impact of the data length of host status history in our algorithms such that they perform the best on four well-known VM selection methods found in the literature. In addition, we investigate how the four VM selection methods have impact on the performance in terms of the energy consumption, the number of SLA violations, the number of migrations, and other metrics.

1.3.3 Minimizing biased VM selection (Chapter 4).

- Propose two VM selection algorithms termed as Minimum VM Migrated Count (MiMc) and Minimum migration time Minimum VM Migrated Count (MmtMiMc) that resolve biased VM selection in live VM migration.
- Propose two new metrics, which are the maximum number of VM migrated count and the degree of load balancing of VMs migrated count.
- A simulation-based evaluation and performance analysis of the algorithms using the real PlanetLab workloads and a random workload.

1.3.4 Proactive selection process across cloud data centers (Chapter 5).

- Modify the system model to support proactive selection process techniques that reduce network reconfiguration problem in WAN live VM migration. This model has been proposed to consider neglected parameters and metrics that have an effect on live migration cost.
- Propose a VM selection algorithm that aims to be a proactive solution for decreasing migration time by minimizing the number of IP reconfigurations that are required in case of WAN migration between the data centers.
- Propose algorithms to find the suitable data center for the placement of the VM selected for migration from the overloaded hosts. This criterion aims to minimize the service downtime.

- Propose new metrics to evaluate WAN live migration cost. We extended CloudSim to simulate and evaluate our algorithms and metrics for VM migration across the data centers on random workload.
- Perform an extensive simulation based evaluation and performance analysis of the proposed algorithms with the well-known VM selection methods.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: In Chapter 2, we present a comparative study of selection approaches used in the live VM migration technique. In Chapter 3, we propose efficient algorithms by studying host detection and VM placement problems. These proposed algorithms consider the trade-off between power consumption and SLA violation. In Chapter 4, we propose algorithms for minimizing biased VM selection in live VM migration. In Chapter 5, we propose a new system model and propose algorithms as proactive criteria for live WAN migration that minimizes the number of the IP reconfigurations and new defined metrics. Chapter 6 provides concluding remarks with a discussion of future works.

Chapter 2

A Taxonomy of Literature Review

2.1 Overview

Live virtual machine (VM) migration is defined as a technique that migrates the entire OS and its associated applications from one host/physical server to another providing that users should not notice any interruption in their services. Live VM migration plays an important role to facilitate online maintenance, load balancing, and energy management as part of resource management. As mentioned before, live VM migration can be divided into two parts: 1) selection process which involves three different phases: when to trigger the migration, which VMs must be selected to be migrated, and which destination host must be chosen to move the selected VMs. 2) migration process that targets moving the VM in minimum time to avoid any interruption of services.

In this chapter, the main selection algorithms for host detection, VM selection, and VM placement are discussed, and some IP network reconfiguration solutions in WAN area migration are reviewed. The selection algorithms will be categorized based on their

methodologies and approaches. In each category, we discuss the algorithm scope along with other performance metrics and the considered parameters in this category.

2.2 Host Detection Approaches

When a host is overloaded one or more VM live migration is required from the host under consideration. In [40] researchers proposed a fixed utilization threshold policy. The policy sets upper and lower utilization thresholds for hosts, and the total utilization of the CPU should be kept between the upper and lower thresholds. If the CPU utilization of a host is less than the lower threshold, the algorithm detects an underutilized host. As a consequence, all VMs have to be moved from this host to another host, and the host has to be turned off. On the other hand, if the utilization is higher than the upper threshold, the host is declared to be in an overutilized situation. As a result, some VMs are migrated to reduce the utilization from this host. The static thresholds values are not a suitable solution in dynamic environment with unexpected workloads.

Authors in [41] proposed the averaging threshold-based algorithm (THR). It computes the mean of the n last CPU utilization values and compares it to the previously defined threshold. The algorithm detects underload state if the average of the *n* last CPU utilization measurements is lower than the specified threshold. This algorithm is unsuitable for a dynamic environment.

In [42] researchers proposed four policies in two categories. The first category is Adaptive utilization threshold based algorithms that include two policies: Median Absolute Deviation (MAD) and InterQuartile Range (IQR). These policies offer auto-adjustment of the utilization thresholds based on a statistical analysis of historical data obtained during the lifetime of the VMs. The objective is to alter the value of the upper

utilization threshold based on the strength of deviation of the CPU utilization. MAD is defined as a measure of statistical dispersion that performs better with distributions without a mean or variance. Also, it is a more robust estimator of scale in comparison to sample variance or standard deviation. The main disadvantage in MAD is that the magnitude of the distances of a small number of outliers is inappropriate. IQR is another measure of statistical dispersion. It is called the midspread or middle fifty which means the difference between the third and first quartiles in descriptive statistics. This category has a poor prediction of host overloading. Moreover, when a host has encountered the same CPU utilizations in the past, the value of the threshold in these approaches is measured around 100%, resulting in a more aggressive consolidation of VMs and more SLA violation.

The second category is regression based algorithms that include two policies: Local Regression (LR) and Local Robust Regression (LRR). These depend on estimation the future CPU utilization. They perform better forecasting of host overloading but has higher complexity. LR is an approach that fits a curve that shows the trend in the data. A host is overloaded in case the maximum migration time is closer than a safety margin to the trend line. LRR compares the maximum migration time to an expected value and weights it before making the decision of overloading in the host. This category is influenced by the presence of outliers and does not reflect the behavior of the bulk of the data.

Researchers in [43] proposed a linear regression-based CPU usage prediction (LiRCUP). This algorithm predicts the future state based on historical data of each host. The algorithm measures the future CPU usage to predict overloaded and underloaded

hosts. This leads that some of the VMs will be moved to other hosts before an SLA violation occurs. As a result, undesirable migrations occur even when the current resource usage of the considered hosts is still acceptable.

In [44] authors proposed Adaptive Migration Threshold algorithm for host detection. The authors do not use the historical data, but they use only the current resource utilization to measure an upper and lower utilization threshold values for each host. The algorithm uses resources utilization, including CPU utilization, RAM and bandwidth, to measure an upper and lower utilization threshold values for each host.

In [45] researchers proposed Dynamic Fuzzy Q-Learning (DFQL) algorithm. This algorithm relies on Fuzzy Q-Learning to detect overloaded hosts. Fuzzy C-Mean (FCM) as a fuzzy clustering algorithm has been applied to estimate Gaussian membership functions. The algorithm selects a new threshold for each host every time based on the performance feedback. The convergence learning time for the algorithm is long.

Authors in [46] proposed an adaptive fuzzy threshold based algorithm. For detecting overload and underload hosts, the algorithm uses the current and estimated resource usage for a more efficient upper and lower threshold values. The algorithm collects the information from a host and feeds them to a fuzzy inference engine supported by the Sugeno fuzzy rule set to determine if the host is overloaded. Also, the lower threshold is dynamically measured according to workload changes.

In [47] researchers proposed a VM-based dynamic threshold (VDT) algorithm to detect underload host. This algorithm computes utilization of host based on considering host CPU utilization and the number of VMs on the host. VDT algorithm selects host as a candidate host for migrating all of its VMs based on the minimum CPU utilization among

hosts. In the case of the CPU utilization between hosts is equal, a host with least number of VMs will be considered as underloaded host.

In [48] authors proposed a multi-criteria technique for detecting underutilized hosts including Available Capacity (AC), Migration Delay (MDL), and TOPSIS-Available Capacity-Number of VMs-Migration Delay (TACND) policies. AC uses available resource capacity as parameter to detect underloaded host. MDL algorithm uses migration delay as a measure to determine underloaded among all candidate hosts. TACND discovers underloaded host based on three criteria, which includes host available capacity, number of VMs on the host, and the migration delays of VMs on the host.

In [49] researchers proposed an approach for deriving an optimal policy for detecting host overloading conditions, which optimally solves the problem of host overloading detection by maximizing the mean intermigration time under the specified QoS goal. One of their assumptions is that the workload satisfies the Markov property, which may not be true for all types of workloads.

Authors in [50] proposed a virtual machine consolidation algorithm with usage prediction (VMCUP). The VMCUP algorithm is executed during the consolidation process to decide when a host is overloaded or underutilized based on the current and future (i.e., predicted) resource usage.

Authors in [51] proposed a modified of five host overload detection algorithms [41, 42] using mean and standard deviation. The algorithms are modified in such a manner that the host will be declared overloaded if the built-in overload detection finds the host overloaded while the requested utilization is higher than the capacity or predicted utilization is higher than the capacity.

Table 2-1 compares existing host detection algorithms concerning the parameters used in the detection process for each technique and shows the metrics considered as well.

Table 2-1: Host Overload Detection State-of-Art Algorithms Comparison

Technique	**Parameters Considered**								**Metrics Considered**		
	Number of VMs on the Host	historical Data	current CPU Utilization	Static Threshold	Dynamic Threshold	Future CPU Utilization	Available Capacity of a Host	Migration Delays of VMs	Power Consumption	SLA Violation	No. of Migrations
Beloglazov et al. [40]			✓	✓							
THR [41]		✓		✓					✓	✓	✓
MAD [42]		✓			✓				✓	✓	✓
IQR [42]		✓			✓				✓	✓	✓
LR [42]		✓		✓		✓			✓	✓	✓
LRR [42]		✓		✓		✓			✓	✓	✓
LiRCUP [43]		✓		✓		✓			✓	✓	✓
Adaptive Migration Threshold [44]			✓		✓				✓	✓	✓
DFQL [45]	✓	✓							✓	✓	✓
fuzzy threshold [46]		✓			✓				✓	✓	✓
VDT [47]	✓		✓		✓				✓	✓	✓
AC [48]							✓		✓	✓	✓
MDL [48]								✓	✓	✓	✓
TACND [48]	✓						✓	✓	✓	✓	✓

In summary, most of the existing detection algorithms are based on the current CPU utilization of the system. If a host is determined to be overloaded at a moment, then VM migration is initiated immediately, which is not the best choice. It should be noted that each VM migration is associated with some performance degradation that in turn increases the SLA violation rate. It is known that there is a strong relationship between

determining when a VM migration should be initiated and the cost associated with extra SLA violation rate.

2.3 VM Selection Approaches

A VMs selection approach is applied after the completion of the host overload detection phase. Three different methods are suggested by researchers in [40, 42] to select the VMs that have to be moved to the underutilized hosts. The first approach is called Minimization of Migrations (MM), in this approach the minimum number of VMs is moved to underload hosts to reduce migration overhead. Descending VMs CPU utilization ordering step is implemented as the first step in this algorithm, after that a repeated scanning for the ordered list is performed to find the best candidate VMs to be migrated. The candidacy of the VMs will be based on the following two conditions.

The first condition to be met is that the VM must have a higher utilization value when compared to the difference between the host's overall utilization and the upper utilization threshold. The second condition that has to be satisfied when the VM is migrated from the host, the difference between the upper threshold and the new utilization should be the minimum of values provided by all the VMs. Then, if a suitable VM is not found, a VM with the highest utilization value is selected to be removed from the list. Iterations are repeated until a utilization value is found which is less than the upper utilization threshold.

The second algorithm is Highest Potential Growth (HPG). This policy migrates VMs which have, relatively, the lowest value of CPU usage to reduce the total likelihood increase of the utilization and SLA violation.

The third algorithm is Random Selection (RS). The algorithm chooses a VM to be moved according to a uniformly distributed discrete random variable whose values index a set of VMs allocated to a host.

Authors in [41] proposed minimum migration time maximum CPU utilization algorithm (MMTMU). The algorithm first selects VMs with the lowest value of RAM to minimize the live migration time. After that, the algorithm selects the VMs from the selected subset that resulted from previous step based on the maximum CPU utilization, by taking the average over the last values to reduce the overall CPU utilization of the host maximally.

Researchers in [51] proposed modified MMT and MC [52] VM selections algorithm by using migration control. No migration will occur in the case of a VM that is steadily occupying high resource of a host for some period.

Two different algorithms have been proposed by authors in [52], named Minimum Migration Time (MMT). In this method, a VM is chosen based on the value of the migration time, the less the better. Migration time can be easily computed as the amount of RAM utilized by the VM divided by the additional network bandwidth available for the host.

The other algorithm is namely known as the Maximum Correlation (MC). In this approach, a correlation value is calculated. Whenever the value of the correlation between the resource usage by applications running on an oversubscribed host then the likelihood of overloading will be higher. So, the selection of the VMs to be migrated is based on the correlation of the CPU utilization with other VMs, the highest correlation

value is selected. To assess the prediction quality of the dependent variable the multiple correlation coefficient is used in multiple regression analysis.

Authors in [53] proposed two algorithms. The first algorithm is called the Median Migration Time (MedianMT). This method selects a given VM that requires the median time to complete a migration relatively to the other VMs allocated to the host. The second algorithm is the Maximum Utilization (MU) that selects a VM to migrate from the overutilized host based on the largest possible usage of CPU can be that expected to minimize the number of migrations.

In [54] authors proposed a modified NVMMP algorithm based on VM priority. The algorithm first sort VMs with the highest CPU utilization value. After that, the algorithm selects the VMs from the selected subset that resulted from the first step based on minimum execution, i.e. their maximum execution left.

In [55] authors proposed Host Fault Detection (HFD) algorithm that selects a VM to migrate from the overutilized host based on the maximum impact on the cause of the overload. If the overload is caused by RAM, then the VM with the maximum allocated RAM is selected by the algorithm.

Table 2-2 compares existing VM selection algorithms concerning the parameters used in the selection process for each technique and shows the metrics considered to evaluate the algorithm as well.

Table 2-2: VM Selection State-of-Art Algorithms Comparison

Technique	Parameters Considered					Metrics Considered		
	VM CPU Utilization	VM RAM Utilization	Host Network Bandwidth	Uniformly Distributed Discrete Random Variable	Execution Time Left	Energy Consumption	SLA Violation	Number VM Migration
MM [40, 42]	✓					✓	✓	✓
HPG [40, 42]	✓					✓	✓	✓
RC [40, 42]				✓		✓	✓	✓
MMT [52]		✓	✓			✓	✓	✓
MC [52]	✓				✓	✓	✓	✓
MedianMT [53]		✓	✓			✓	✓	✓
MaxU [53]	✓					✓	✓	✓
modified NVMMP [54]	✓				✓			✓
modified of MMT [51]		✓	✓			✓	✓	
modified of MC [51]	✓							
HFD [55]	✓	✓				✓	✓	✓
MMTMU [41]	✓	✓	✓			✓	✓	✓

In summary, the existing VM selection algorithms focused on minimizing the number of VM migrations and on reducing performance degradation. It should be noted that no proactive criteria exist for live WAN migration that minimizes the number of the IP reconfigurations. It is known that if the time needed for IP reconfiguration for all migrated VM users increases, then there will be an increase in the interruption of service, network overhead and performance degradation.

2.4 VM Placement Approaches

VM placement is the process that comes after the completion of the VM selection phase. In this section, we discuss these algorithms. In addition, we discuss other VM placement algorithms perspectives, such as VM placement resulting from a new user request to

create a new VM in a suitable host, or VM placement caused by dynamically reassigning VMs to hosts due to changes of system conditions or VM requirements.

Average traffic latency reduction is the objective that researchers in [56] is concentrating on. To achieve this objective a traffic aware VM placement algorithm has been proposed. Two traffic models have been proposed, which are namely known as, partitioned and global. In partitioned model the only allowed communication is the one between the VMs in the same partition. Whereas in the global traffic model, the communication is not constrained on the VMs in the same partition with a constant flow rate. In this algorithm, better network scalability is satisfied by reducing the traffic going through the switches. This can be explained by the fact that this algorithm is moving the VMs through a minimum number of switches.

In [57] authors formulated the VM placement problem as a multi-objective optimization problem to minimizing total resource wastage, energy consumption, and thermal dissipation cost. The authors proposed an improved genetic algorithm with a fuzzy multi-objective evaluation to search for solutions for allocating VMs.

In [58] researchers proposed a VM placement algorithm based on the Ant Colony Optimization (ACO) meta-heuristic where the placement is computed in a dynamic way according to the current load by modeling the workload consolidation problem as an instance of the Multidimensional Bin-Packing (MDBP) problem. The goal of this algorithm is to pack the VMs into fewer hosts. The algorithm needs the knowledge about all the workload and its related resource requirements to compute the placement.

Authors in [59] formulated the VM placement problem as a multi-objective ACO algorithm to minimize SLA violation, total resource wastage, and power consumption. In

ACO algorithm, each ant constructs a solution for selecting VM to the target server. The constructed solution is estimated by the suitable function, which combines SLA violation, resource consumption, and power consumption.

In [60] authors proposed a joint energy-aware and application aware VM placement strategy based on the theory of multi-objective optimization by exploring a balance between server energy consumption and the communication network energy consumption in the data center. The algorithm aims to meet the conditions of the server-side constraints, to minimize network data transmission, and to reduce power consumption in data centers. The considered parameters are the distance between the switches that interconnect physical hosts, constraints of servers and the application dependencies among VMs of composite applications.

In [61] researchers applied the genetic algorithm to address the VM placement problem considering reducing energy consumption and the communication network among hosts. The authors present a VM placement model considering two functions. The first function is a linear function of its workload that shows the energy consumed by a server and the energy consumed when the server is idle. The second one is a function of the amount of data exchanged among the VMs that displays energy consumed by the network.

In [62] authors proposed a hybrid genetic algorithm (HGA). The HGA algorithm approach is used to allocate VMs efficiently than the genetic approach in [61]. A repairing procedure is embedded for converting the proposed solution into a feasible one. This can be accomplished by the means of local optimization procedure and resolving the existing violations in order to improve the overall quality of a solution.

Researchers in [63] proposed Family Genetic Algorithm (FGA) for VM placement to overcome the limitations of the Genetic approaches [61, 62]. These limitations are the premature convergence and the high processing time.

In [64] researchers proposed VM Scheduler placement algorithm to reduce the time of allocation of VM to the server and to optimize the resource utilization. The algorithm represents the list of resources in a binary search tree (BST) instead of representing them in a queue. The algorithm generates two BSTs, one for VM specification and one for hosts. The VM scheduler takes the VM that has the maximum requirement and searches for a candidate host which best fits the requirement of VM.

In [42, 52, 67] authors proposed Power Aware Best Fit Decreasing (PABFD) algorithm for VM placement to move the VMs from the overloaded host to unloaded host or from underloaded host for server consolidation. After sorting all migrated VMs based on a VM selection method, the algorithm selects the destination host to receive the migrated VM, which causes the least increase in the power consumption. The algorithm relies on the traditional greedy algorithm to optimize the allocation of VMs.

In [47] researchers proposed host utilization and minimum correlation (UMC) VM Placement Algorithm to reallocate VMs from overutilized hosts or from underutilized host. The considered parameters are host utilization and the correlation between the resources of a VM with the VMs present on the host correlation. The algorithm selects the destination host to receive the migrated VM if its CPU utilization has the lowest correlation with all VMs CPU utilization on that host.

Table 2-3 compares existing VM placement algorithms concerning the parameters used in the allocation process for each technique and shows the metrics considered as well.

Table 2-3: VM Placement State-of-Art Algorithms Comparison

Technique	Parameters Considered							Metrics Considered							
	VM Spec.	Host CPU	Host RAM	Network Switches	Network Link	VMs Application Dependencies	Network B W.	Number of Active Host	Execution Time	Resource Wastage	Number of Switches	Number of Migration	Energy Consumption	SLA Violation	Temperature
Meng et al. [56]				✓	✓						✓		✓		
Xu et al. [57]		✓	✓							✓			✓		✓
Feller et al. [58]		✓	✓				✓	✓	✓						
Fei Ma et al. [59]	✓	✓	✓				✓			✓			✓	✓	
Huang et al. [60]						✓		✓					✓		
Wu et al. [61]		✓						✓	✓				✓		
HGA [62]							✓	✓					✓		
FGA [63]							✓	✓					✓		
Mandal and Khilar [64]	✓								✓	✓					
Beloglazov and Buyya [42, 49]		✓										✓	✓	✓	
Horri et al. [47]		✓				✓						✓	✓	✓	

In summary, the existing VM placement approaches focused on reducing the number of physical machines, VM allocation time and the data center energy consumption. It should be noted that no proactive criteria exist for live WAN migration that minimizes IP reconfiguration time, which results in minimizing the service downtime. It is known that if the time needed for IP reconfiguration for all migrated VM

users increases there will be an increase in interruption of the service time, network overhead and performance degradation.

2.5 WAN Area Migration Solutions

Over the last two decades, there has been significant research to migrate the VMs to different data centers that are located at different geographic locations (i.e. different subnet configurations) to obtain high QoS. Thus, WAN live VM migration techniques have been proposed [16, 29-38, 65]. There are a few techniques proposed to solve IP network reconfiguration [29, 30, 33, 36-38].

Bradford et al. [29] proposed a solution that depends on DNS resolutions to transfer on going network connections transparently. When VMs migrate, they maintain their canonical names, and the new IP address is registered with the named host. Lookups for the VM based on the canonical name, following migration, will resolve to the new (correct) IP address. This seamless change in original IP address and resolution of new IP address while the VM migrates across different networks is done through IP tunneling. Tunneling is a mechanism for providing a path to networks/LANs of different IP configurations by taking help from the gateways encountered on the way to the destination network (where the designated host resides). Gateways provide tunnel endpoints, preventing any average loss of connectivity. Note that this solution places the burden of managing endpoints on the applications (i.e., they need to be aware of the IP address change).

Silvera et al. [31] proposed not to change IP address of the virtual machine while being migrated between different subnets. Agents on the source and destination subnets

are responsible for ensuring the continued connectivity of the virtual machine via the use of Proxy-ARP. IP-in-IP tunnels are used between the subnet agents to forward between subnets the traffic destined to/originating from the VM.

Wood et al. [30] proposed a combination of layer 3 virtual private networks (VPNs) and layer 2 virtual private LAN service (VPLS) to provide end-to-end routing across multiple networks and bridge LANs at different locations. The unified virtual network provides the view of a LAN to migrating VMs, with VMs maintaining single IP address.

The above methods [29-31] do not support the establishment of a new TCP connection in conjunction with VM migration, which causes increased network delay time and traffic congestion and increased performance degradation.

Kuribayashi et al. [38] proposed mSCTP, which supports multihoming and multiple IP addresses simultaneously. In mSCTP-based migration, VMs will transfer data using different TCP connections before and after migration, which causes this feature to improve response time and enhance throughput.

In summary, the existing techniques focused on applying a mobility solution or scheme to maintain the network connectivity and to preserve the open connections during and after the migration in the migration process. But no proactive criteria exist for live WAN migration that minimizes the number of the IP reconfigurations. It is known that if the time needed for IP reconfiguration for all migrated VM users increases, then there will be an increase in the service interruption time, network overhead and performance degradation.

2.6 Summary

In summary, most of the existing detection algorithms are based on the historical data of the system, which is not always a good indicator on the real workload on the VM, especially when we have a dynamic environment with unpredictable workloads. our work should be concentrating on finding the optimal value for the detection threshold to trigger the detection process. It should be noted that each VM migration is associated with some performance degradation that in turn increases the SLA violation rate. It is known that there is a strong relationship between determining when a VM migration should be initiated and the cost associated with extra SLA violation rate and energy consumption.

The existing VM selection and VM placement approaches focused on minimizing the number of VM migrations, reducing performance degradation, reducing the number of physical machines, VM allocation time and the data center energy consumption. It should be noted that no proactive criteria exist for live WAN migration that minimizes the number of the IP reconfigurations. It is known that if the time needed for IP reconfiguration for all migrated VM users increases, then there will be an increase in the interruption of service, network overhead and performance degradation.

Chapter 3

Markov Prediction Model for Host Load Detection and VM Placement

3.1 Overview

The design of good host overload/underload detection and VM placement algorithms play a vital role in assuring the smoothness of VM live migration. However, the existing algorithms have some shortcomings when it comes to the prediction of the future load state for the VMs. The presence of the dynamic environment that leads to a changing load on the VMs motivates us to propose a novel Markov prediction model to forecast the future load state of the host. We propose a host load detection algorithm to find the future overutilized/underutilized hosts state to avoid immediate VMs migration. Moreover, we propose a VM placement algorithm determine the set of candidates hosts to receive the migrated VMs in a way to reduce their VM migrations in near future. We evaluate our

proposed algorithms through CloudSim simulation on different types of PlanetLab real and random workloads. The experimental results show that our proposed algorithms have a significant reduction in terms of SLA violation, number of VM migrations, and other metrics than the other competitive algorithms.

3.2 Proposed Markov Host Prediction Model

This section explains the forecasting model used to effectively decide whether it is really necessary to migrate a VM depending on the present as well as the predicted future load based on previously observed values using Markov model prediction technique [66].

In the Markov chain, the observed variable W is discretized, so the observation sequence $w_1, w_2, \dots, w_n$ can be described using a discrete scalar observation sequence $\{w_1, w_2, \dots, w_n\}$ as proposed in our forecasting model, the last w observations of a given host CPU utilization, where each of the variables w_n may take one of M different states $\{S_1, S_2, \dots, S_M\}$. In our proposed Markov model, in the proposed algorithms, three different states for a given host are possible, namely underloaded (U), normal loaded (N) and overloaded (O).

The Markov model will be used to model the host detection depending on historical data that will be maintained in a log file. The historical training set is stored in a database, in our forecasting model the prediction will take place when we have at least 10 historical data observations stored in the database. This number is used in other algorithms [42, 52, 67]. The Markov model is built using three states given by $\{S_1 = U,\ S_2 = N,\ S_3 = O\}$. The stochastic variable χ is a discrete random variable taking one of these three values,

where Algorithm 2-1 will be applied periodically by each host manager to find χ for each observation and register it in the host log file.

Algorithm 2-1: Host Detection State.

1 ***Input:*** *host CPU utilization of host j (*$CPUu(H_j)$*, lower threshold, and upper threshold.*
2 **Output:** χ (current host state).
3 $\boldsymbol{If}\ \ CPUu(H_j) \leq lower\ threshold\ \boldsymbol{then}$
4 $\quad \chi \leftarrow U$
5 $\boldsymbol{else\ If}\ lower\ threshold < CPUu(H_j) < upper\ threshold\ \boldsymbol{then}$
6 $\quad \chi \leftarrow N$
7 $\boldsymbol{else\ If}\ CPUu(H_j) \geq\ upper\ threshold\ \boldsymbol{then}$
8 $\quad \chi \leftarrow O$
9 $\boldsymbol{return}\ \chi$

Algorithm 2-1 shows the pseudo-code of host detection state for each observation. Three parameters are inputs to this algorithm. The first parameter is the host CPU utilization, which is calculated by dividing the total MIPS requested on the total host MIPS. The other parameter is the lower threshold, which is assigned a value of 0.1. The upper threshold is taken from the MAD algorithm which is explained in [52]. Each host has an underloaded (U), over- loaded (O) or normal loaded (N) state, which can be easily found by comparing the current CPU utilization value ($CPUu$) by the lower and upper thresholds. After the host load state is determined it is stored in the log file in order to be used in our proposed Markov prediction algorithm.

It should be noted that the first-order Markov chain is most widely used in describing dynamic processes, wherein the conditional probability of an observation w, at time n (i.e., w_n) only depends on the observation, w, at time $n-1$ (i.e., w_{n-1}) as shown in Equation (3.1). Moreover, the joint probability of n observations, $P(w_1, w_2, \dots, w_n)$

using the first order Markov chain can be given by Equation (3.2). Our Markov detection algorithm starts working after collecting 10 historical observations (n = 10).

$$P(w_n \mid w_{n-1}, w_{n-2}, \dots, w_1) \approx P(w_n \mid w_{n-1}) \tag{3.1}$$

$$\mathrm{P}(w_1, \dots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1}) \tag{3.2}$$

where the conditional probabilities $p(w_n = S_j \mid w_{n-1} = S_i)$ are referred to as state transition probabilities or simply transition probabilities. The transition probabilities describe the probability of the system at state S_j at time n given that the system was at state S_i at time $n-1$. In most cases, we assume that the transition probabilities are homogeneous, which means that the probabilities do not change over time, so

$$\begin{aligned} p(w_n = S_j \mid w_{n-1} = S_i) \\ = p(w_{n+T} = S_j \mid w_{n-1+T} = S_i) \end{aligned} \tag{3.3}$$

where T represents a positive integer larger or equal to one. The transition probabilities can be written as a transition matrix, which is of dimension $M * M$ for a system with M ($where\ M = 3$) different states $\{S_1, S_2, \dots \dots, S_M\}$.

The state and transition probabilities of a given Markov chain can be shown using graph. Figure 3-1 shows our host detection Markov model with three discrete states $\{O, \mathrm{U}, \mathrm{N}\}$ with every periodic time we would transit to a (possibly) new state based on the probabilities in Equation (3.4). The system model starts in one of these states and moves successively from one state to another. Each move is called a step. The probability p_{ij} represents the chance of the system model to be in the current state S_i and moves to next state S_j.

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1M} \\ p_{21} & \mathrm{p}_{22} & \cdots & p_{23} \\ \vdots & & \vdots & \\ p_{M1} & p_{M2} & & p_{MM} \end{bmatrix} = \begin{matrix} \\ U \\ N \\ O \end{matrix} \begin{matrix} \begin{matrix} U & N & O \end{matrix} \\ \begin{bmatrix} p_{UU} & p_{UN} & p_{UO} \\ p_{NU} & p_{NN} & p_{NO} \\ p_{OU} & p_{ON} & p_{OO} \end{bmatrix} \end{matrix} \quad (3.4)$$

Since each element in the matrix represents a probability of staying or moving to another state, so the matrix element of a given row should satisfy the following properties

$$p_{ij} = p(j|i) \geq 0, for\ all\ i, j \quad (3.5a)$$

$$\sum_{j=1}^{M} p_{ij} = 1, for\ i \in \{1, \ldots, M\} \quad (3.5b)$$

The state and transition probabilities of a given Markov chain can be shown using graph. Figure 3-1 shows our host detection Markov Model with three discrete states $\{O, \mathrm{U}, \mathrm{N}\}$. and every periodic time we would transit to a (possibly) new state based on the probabilities in Equation 3.4. The system model starts in one of these states and moves successively from one state to another. Each move is called a step. The probability p_{ij} represents the chance of the system model to be in the current state S_i and moves to next state S_j.

Instead of immediately migrating some of its VMs we can check whether the migration is required or not. The algorithm takes states and transition probabilities of a given host j detection from Markov model as an input and makes the decision of migration and the decision of hosting VMs as an output. The decision is based on the current CPU utilization and the future CPU utilization.

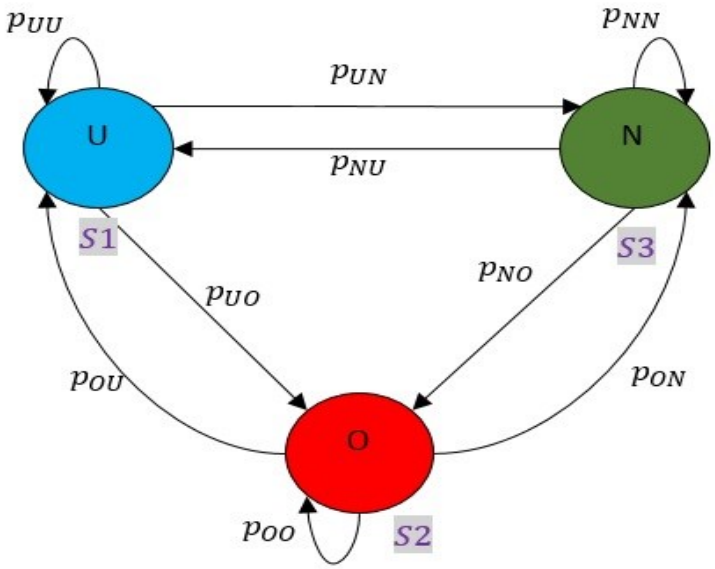

Figure 3-1: States and Transition probabilities of the Host detection Markov Model

3.3 Proposed System

In this section, we present a Markov-based host detection and VM placement algorithms for cloud data center. In section 3.3.1, our proposed system architecture is explained, underload/overload detection algorithm is then explained. Then a VM placement algorithm is explained. Finally, illustrative scenarios are clarified.

3.3.1 System Architecture

The target system is an IaaS environment, represented by a large-scale data center. The data center consists of less than J heterogeneous hosts where each host contains multiple VMs. Multiple VMs can be allocated to each host through VMM. Besides, each host and VM are characterized by the CPU performance metrics defined in term of Millions Instructions Per Second (MIPS), the amount of RAM and network band-width. The target system model is depicted in Figure 3-2 which is a modified version of the model

described in [68]. Our model includes two important parts: A Data Center manager that has an extra predictive VM placement functionality, and the Host Manager that has an extra Markov model prediction agent for host detection.

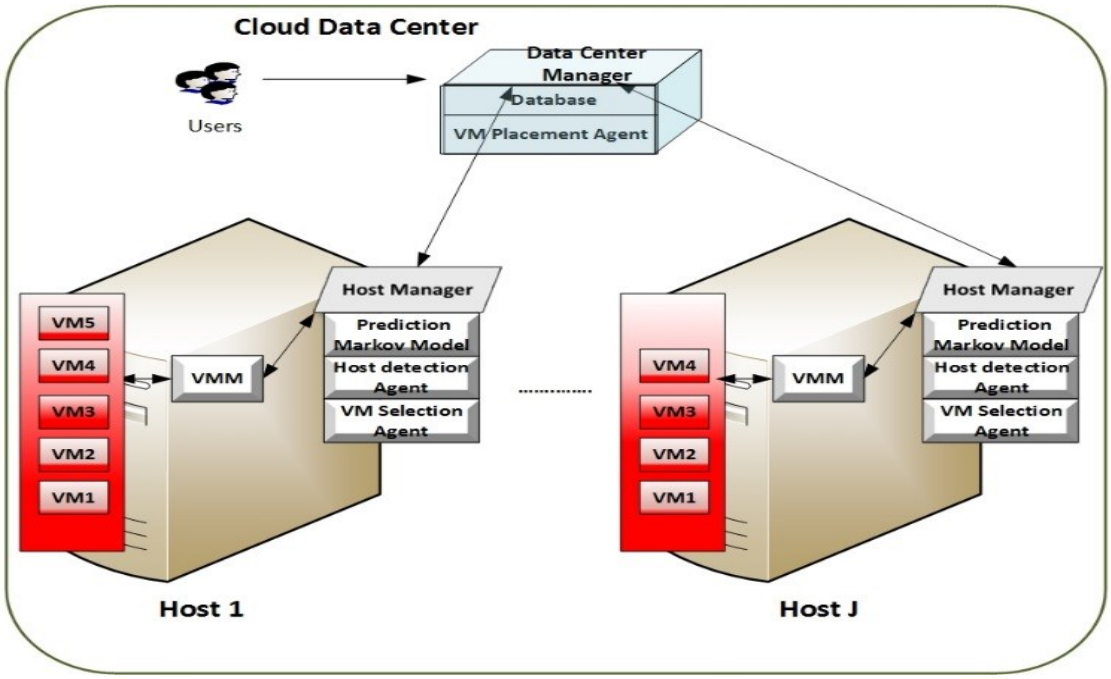

Figure 3-2: System Model

Figure 3-2 shows the Host Manager and the Data Center Manager components. Host manager resides on every host for keeping continuous observation on CPU utilization of the node. Data center manager interacts with the host managers.

Host Manager consists of the following components:

- Host detection agent: responsible for detecting the current load state of the host, which can be either underloaded or overloaded.
- VM selection agent: responsible for finding the VM that has to be migrated.
- Prediction Markov model: responsible for finding the future load state of the host.

- VMM: responsible for monitoring host as well as sending gathered information to the data center manager. In addition, VMM performs actual resizing and migration of VMs as well as changes in power modes of the PMs.
- Data Center Manager consists of the following components:
- VM placement agent: responsible for performing the migration from overloaded/underloaded hosts to the candidate hosts based on a predictive Markov model.
- Database: data structure that contains all the information about the hosts and the utilization of each host.

Our proposed algorithms suggest that the load state host detection algorithm and the VM placement algorithm should not only depend on the current overall rewards gained from migrating the VMs, but also the future rewards should be taken into consideration for better SLA violation, and number of VM migrations. Host manager interacts with the VMM manager in order to initiate the VM migration process after finishing the host detection, and VM selection processes. It also interacts with the data center manager in order to initiate the VM placement.

The host manager is interacting with the detection Markov model, which is shown in more details in the sequence diagram below. Our proposed algorithms suggested that the load state host detection algorithm and the VM placement algorithm, should not only depends on the current overall rewards gained from migrating the VMs, but also the future rewards should be taken into consideration for better SLA violation, number of VM migrations. Host manager is interacting with the VMM manager in order to initiate the VM migration process after finishing the host detection, VM selection and VM

placement processes. It also interacts with the data center manager in order to initiate the VM placement.

3.3.2 The Proposed Work

The problem of VM migration can be divided into four parts: (1) determining which hosts are overloaded, thus one or more VM migration is required from the host under consideration, (2) determining which hosts are underloaded so that all VMs should be migrated from those hosts; (3) selecting VMs that should be migrated from overloaded hosts. (4) finding new placement for the migrated VMs by choosing the good candidate hosts [66]. We have proposed three algorithms which resolve the first, second and fourth issues of migration. For VM selection multiple selection algorithms given in [67] are used.

3.3.2.1 Host underload/overload detection

Algorithm 2-2 describes the host overload/underload detection mechanism. Upper and lower thresholds for CPU utilization are assigned first. These can be assigned either statistically using First Order-Markov Chain Host State Detection algorithm (FOMCHSD) or dynamically using Median Absolute Deviation Markov Chain Host Detection algorithm (MadMCHD). In MadMCHD, Median Absolute Deviation (MAD) algorithm is used, which is based on statistical analysis of historical data collected during the lifetime of VMs [42]. For a univariate data set $w_1, w_2, \ldots, w_n$, the MAD is defined as the median of the absolute deviations from the median of the data set:

$$MAD = median_i(\left| w_i - median_j(w_j) \right|) \quad (3.6)$$

The MAD is the median of the absolute values of deviations (residuals) from the data's median. In the proposed overload detection algorithm, the upper CPU utilization threshold (Tu) is defined as given in Equation (3.7)

$$Tu = 1 - s * MAD \quad (3.7)$$

where $s \in R^+$ represents a parameter of the method defining how strongly the system tolerates host overloads. In other words, the parameter s allows the adjustment of the safety of the method: a lower value of s results in a higher tolerance to variation in the CPU utilization.

After our algorithm is triggered, the first thing to calculate is the current CPU utilization, and then to determine whether the static or dynamic values are considered for the upper and the lower threshold by checking the value of the input parameter B. As mentioned before, the values of the upper and lower values are assigned statically or dynamically using MAD. In case B = FOMCHSD, the lower threshold value is equal to 0.1 and the upper threshold value is 0.9. FOMCHSD is a static algorithm.

In case B = MadMCHD the value of the lower threshold is also equal to 0.1 and the value of the upper threshold is calculated using equation in line 12. Our proposed algorithm is triggered when the length of the history data stored in the log file is more than 10.

The current host load state is determined by comparing the value of the current utilization with the lower and the upper threshold. The future load state is predicted using our Markov prediction model. If the future predicted load state is overloaded, then the $migration\_decision\_overloaded$ is assigned a true value and the host is considered for migration. For the underload host detection, if the current state and the future state is

underloaded then the $migration\_decision\_\ underloaded$ is assigned a true value and the host is considered for energy saving or to receive migrated VMs.

Algorithm 2-2: Overload/Underload host detection.

```
1   Input: host, lower threshold = 0.1, upper threshold = 0.9,  B
    (FOMCHSD or MadMCHD).
2   Output:  migration_decision_underloaded  (T/F),
    migration_decision_overloaded (T/F).
3   migration_decision_underloaded ← false
4   migration_decision_overloaded ← false
5   while hostactive = true do
6     if logfile.Length >= 10 then
7     //calculate current CPU utilization of host h
8     utilization ← total Req.Mips/ Total host Mips

9     Switch(B)
10    Case FOMCHSD: break;
11    Case MADMCHD:
12          upper threshold ← 1 − s ∗ MAD

13    //find current utilization using Algorithm 2-1
14    current state ←
      host_detection_state (utilization, lower threshold,
                                          upper threshold)

15  //find future utilization using Markov prediction technique
16  future state ←
    future_Markov_utilization_state(current state)
17  If future state = 0 then
18    migration_decision_overloaded ← True
19  else If current state = U and future state = U then
20    migration_decision_underloaded ← True

21  return migration_decision_underloaded,
          migration _decision_overloaded
```

3.3.2.2 VM placement

VM placement algorithm is the last phase that comes after the detection of the overloaded/underloaded hosts and after the suitable VMs are selected to be migrated.

During this phase, suitable hosts are to be found to migrate all the selected VMs, which fits the requirements of these VMs. In the literature, a single build in VM placement exists in CloudSim [52, 67] called Power Aware Best Fit Decreasing (PABFD), where all the VMs are sorted based on their current CPU utilization in a descending order. Each VM is allocated to a host with the least increase of the power consumption caused by the allocation. We have modified the existing VM placement algorithm by adding the Markov prediction model into the PABFD. In our Markov Power Aware Best Fit Decreasing (MPABFD) algorithm, the future host load state is predicted based on the historical data collected and stored in the log file.

Algorithm 2-3: Markov Power Aware Best Fit Decreasing (MPABFD) algorithm

```
1   Input: hostlist, selected_vm.
2   Output: a host to receive the selected VM
3   minPower ← MAX
4   allocated_host ← None
5   foreach host in hostlist do
6     If (host has enough resources for the selected_vm &&
          hostisactive = true && hoststateafterallocation ()
          ! = O) then
7       TempHostlist1[] ← add.host

8   foreach host in TempHostlist1[] do
9     future state
        ← host.future_Markov_utilization_state(state)
10      If (future state == U or N)then
11        TempHostlist2[] ← add.host

12  foreach host in TempHostlist2 do
13      power ← estimatePower(host, selected_vm)
14      If (power < minPower) then
15        minPower ← power
16        allocated_host ← host

17  return allocated_host
```

Algorithm 2-3 describes our MPABFD algorithm that results in a host to receive the selected VM. The resource availability is first checked for all the active hosts, bearing in mind that all the candidate hosts are not overloaded after the allocation. The candidate temporary host list is stored in array $TempHostlist1[]$. Next the future state for all the candidate hosts stored in $TempHostlist1[]$ are checked. If the future state is overloaded, then the host is excluded from the array. A new temporary array, $TempHostlist2[]$, is generated, which is a subset of $TempHostlist1[]$. Finally, power constraint is considered where a host with the minimum power has higher priority to be selected.

3.3.3 Sequence Diagram Scenarios

In the following sequence diagrams, our proposed algorithms are explained in detail. As previously mentioned, the selection process is gone through three different steps which are the host detection, VM selection and the VM placement. At the very beginning, the host detection agent resides in each host trigger the host detection process, as explained in algorithm.

Figure 3-3 shows an example of Overloaded detection. An overloaded host load state is discovered by the host detection agent and the state is sent to the host manager. The host manager sends the current host load state to the Markov agent to check the future state. As shown in the sequence diagram, the current host load state is overloaded, the Markov agent predicts the future host load state and send it back to the host manager. If the future host load state is either underloaded or normally loaded, then the host manager will ignore the overloaded host and consider it as an underloaded host since it is predicted to be underloaded in the near future.

The host load detection is run repeatedly until the host manager and the Markov agent both determine the host load state as an overloaded. At this specific time, the data center manager is notified of the current/future host overloaded state and the resource availability is checked in the database. Data center manager and host manager are both notified whenever the resources are available. Then the host manager ordered the VM selection agent to find the VM to migrate into it. Placement agent is also notified to find the suitable host for the VM placement to take place. The first thing to be checked by the placement agent is the load state after the VM allocation is performed and whether the state is moved to the overloaded state or not. If the state is going to be overloaded after the allocation process, then the placement is not executed. In the VM placement algorithm, current and future host load state are both considered to make the placement decision. Whenever the current and future host load state is underloaded/normal the VM placement process is started and finally the VM live migration process after checking all the candidate hosts and find the most suitable host with the minimum power. In our scenario, Host 3 and Host 5 both satisfied the first two conditions which are related to the current load state after allocation and the future load state. Host 3 is then selected since it has the minimum power and the VM 3 is migrated to this host.

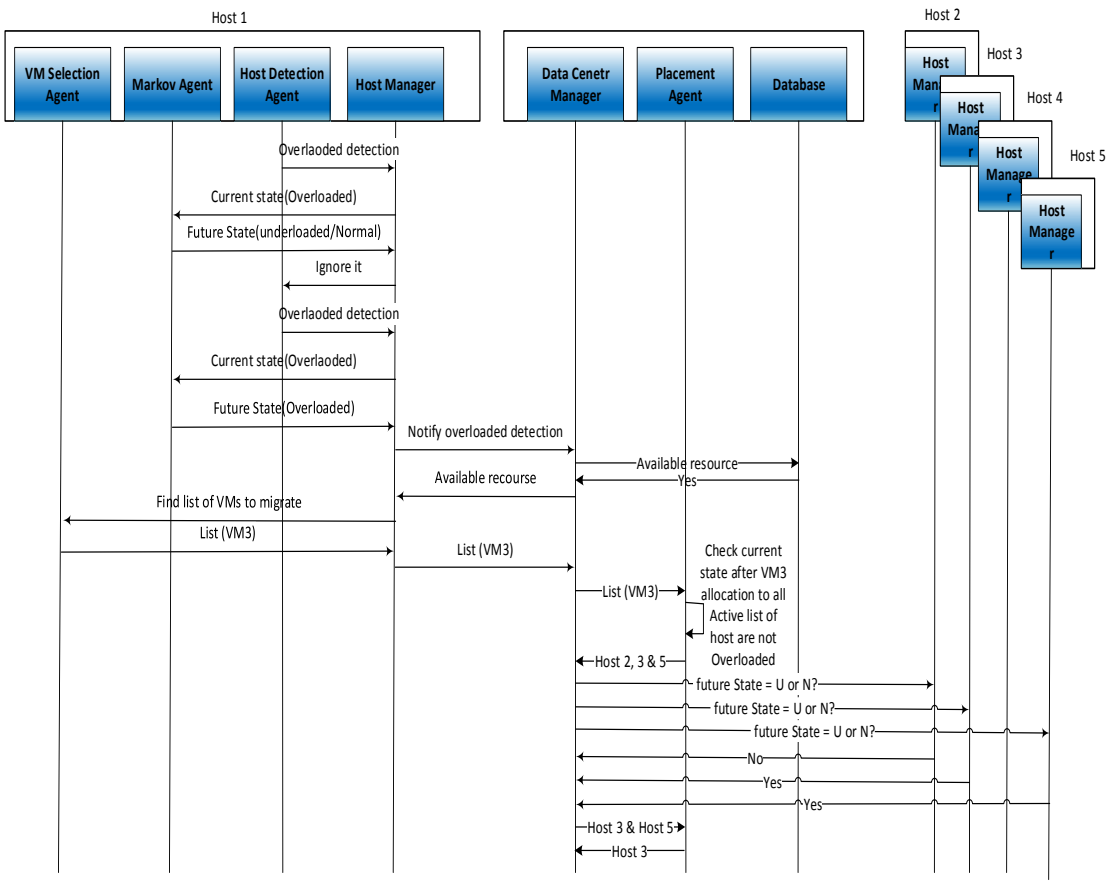

Figure 3-3: Overload Host Detection

On the other hand, an underloaded host load state is discovered by the host detection agent and the state is sent to the host manager. Then Markov agent is the entity responsible on finding the future host load state, after receiving the current host load state. In this scenario, the current host load state is underloaded, and the predicted future load state is either overloaded or in normal state. In this case, the current underloaded state will be neglected since the future state is not underloaded. This checking process is performed periodically by the host detection agent until the current load state matches the future load state, which must be “underloaded”, then the data center manager is informed of the underloaded host state. Resources availability are checked afterwards in order to

perform the migration of the underloaded VMs and shut down those hosts to save power. Whenever an available resource is found, the list of VMs input is fed up to the data center manager and to the placement agent. The list of VMs are sorted in a decreasing order based on the power. In our example, Host3 is considered the best host suitable for the migration since it has the least power and lowest load compared to the other hosts, but before the migration started another metric should be considered which is the future load state for the host. If the future load state is predicted to be overloaded then another host should be found, otherwise the migration process started. This is shown in Figure 3-4 below.

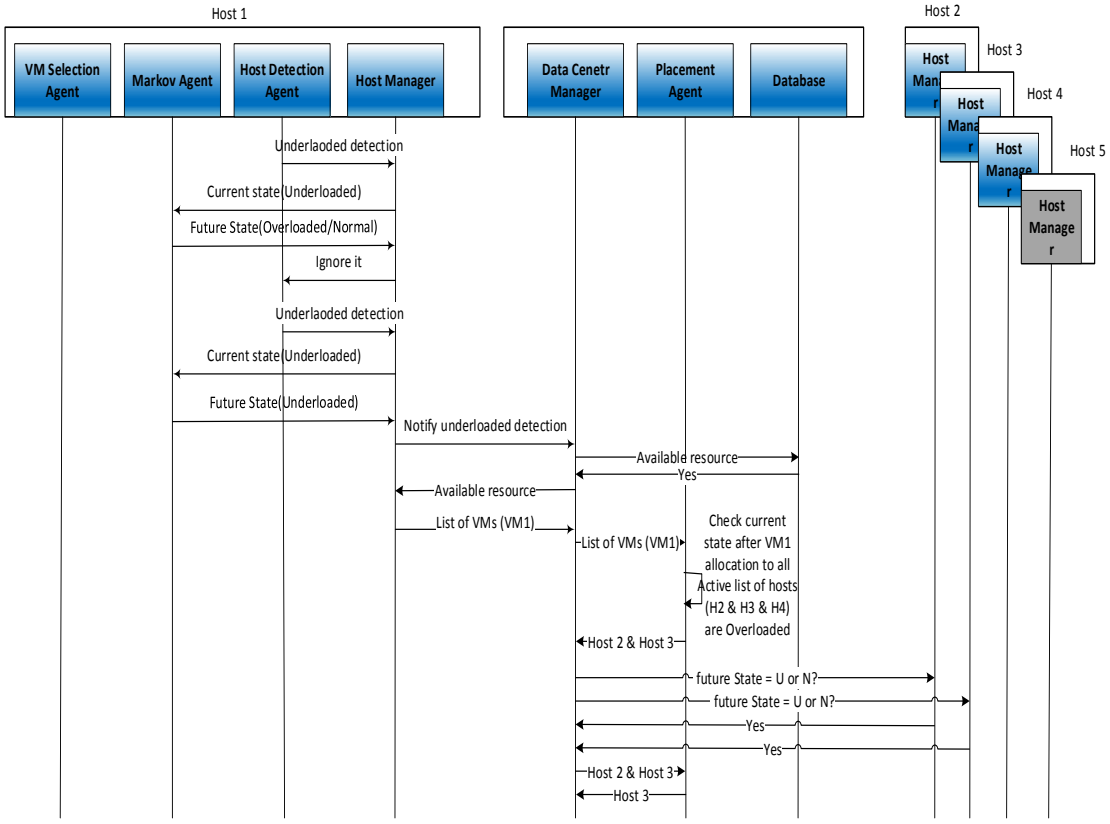

Figure 3-4: Underload Host Detection

3.3.4 Illustrative Scenario

Consider 3 heterogeneous hosts $h = < h1, h2, h3 >$ and 7 VMs $V = < v1, v2, v3, v4, v5, v6, v7 >$ allocated on them. The loads of VMs are allocated to each host as following, $h1 = < v1 = 0.4,\ v2 = 0.2,\ v3 = 0.3 >$, $h2 = < v4 = 0.1 >$, $h3 = < v6 = 0.2,\ v7 = 0.2 >$. The upper threshold is assumed to be a dynamic threshold, $UT\ (h1, h2, h3) = < 0.8, 0.7, 0.7 >$. In addition to the VM loads, each host has extra loads equal to $EL(h1, h2, h3) = < 0.03, 0.04, 0.05 >$.

Host 1 detection agent determines an overload situation has occurred according to:

$$h1_{load} = v1_{load} + v2_{load} + v3_{load} + h1_{El}$$

$$= 0.4 + 0.2 + 0.3 + 0.03 = 0.93.$$

$$h1_{load} \geq h1_{UT} = 0.93 \geq 0.8$$

The aim is to migrate a VM in order to avoid SLA violation. To check the host load future state before migrating some VMs, Markov prediction model agent will calculate the future state using the historical data in h1 given by Historical (h1) $= < u, u, u, o, u, n, n, n, n, u, u, u, u, u, n, o, n, o, o, o >$. The future host load state is calculated as:

$$P(w_n = O \mid w_{n-1} = O) = P_{OO} = \frac{P(w_n = O, w_{n-1} = O)}{P(O)}$$

$$= \frac{P(w_n = O, w_{n-1} = O)}{P(w_n = O, w_{n-1} = U) + P(w_n = O, w_{n-1} = N) + P(w_n = O, w_{n-1} = O)}$$

$$P_{OO} = \frac{2}{1 + 1 + 2} = 0.5$$

Similarly, $P_{OU} = 0.25$ and $P_{ON} = 0.25$. Note that the host will probably stay in the overload situation, therefore some VMs should be migrated. Let the selection agent select

$v2$ to be migrated. To find the destination host for allocating $v2$, MPABFD starts to investigate the first condition, to find the candidate hosts with the capacity requirement still under the threshold after allocating $v2$ as:

$$h2_{Newload} = h2_{load} + v2_{load} = 0.14 + 0.2 = 0.34$$

$$h3_{Newload} = h3_{load} + v2_{load} = 0.45 + 0.2 = 0.65$$

As noted, both new loads are less than their upper thresholds. The second condition is now investigated on both the candidate hosts to predict the future state using their historical data. Considering Historical (h2) $= < u, o, u, o, u. n. n. n. n, u, o, u$ $, u, o, n, o, o, o, o, u >$, we calculate $P_{UU} = 0.1667, P_{UN} = 0.1667$ and $P_{UO} = 0.6667.$ Host 2 will move to overloaded state. Similarly considering Historical (h3) $= <$ $u, u, n, o, n, n, u, n, n, n, o, o, n, n, n, u, n, n, o, n >$, we calculate $P_{NU} = 0.1818$, $P_{NN} =$ 0.5454 and $P_{NO} = 0.2727$. Host 3 will stay in the normal state. It is therefore recommended VM $v2$ to move to host 3 in order to reduce the number of VM migrations and to avoid the SLA violation in the future.

3.4 Experimental setup

In this section, we describe the simulation setup of our proposed approach. We explain the two types of workloads, PlanetLab called a real workload, and random workload. Finally, the evaluation metrics will be described.

3.4.1 Simulation setup

It is difficult to do experiments in a very noticeably dynamic environment like cloud because using real test delimits the experiments to the scale of the infrastructure and

makes reproducing the results an extremely difficult undertaking [69]. In addition, measuring performance in real cloud environment is very sophisticated and time-consuming [70]. For these reasons, the CloudSim simulation tool has been chosen to test our approaches before deploying them in real cloud. Other simulators like GangSim, SimGrid, GridSim [71-73] do not provide suitable environment that can be directly used for modeling cloud computing environment. They are unable to isolate the multilayer service abstractions i.e. SaaS, PaaS and IaaS required by Cloud. On the other hand, The CloudSim tool supports modeling and simulation of data centers on a single physical computing node that contains implemented algorithms in order to compare them with the proposed approach.

To evaluate the efficiency of our algorithms with the existing algorithm, we have used the same experiment setup as used in [41] with some different workload. A data center has been simulated having J heterogeneous physical hosts and V virtual machines. The value of J and V depends on the type of workload which is specified in Table 3-1 [74]. In each workload, half of hosts are HP ProLiant ML110 G4 servers 1,860 MIPS each core, and the other half consists of HP ProLiant ML110 G5 servers with 2,660 MIPS each core. Depending on the CPU and memory capacity four types of single-core VMs are used: High-CPU Medium Instance: 2500 MIPS, 0.85 GB; Extra Large Instance: 2000 MIPS, 3.75 GB; Small Instance: 1000 MIPS, 1.7 GB and Micro Instance: 500 MIPS, 0.633 GB. The characteristics of these VM types are similar to Amazon EC2 instance types.

Table 3-1: Characteristics of the workload data (CPU utilization)

Workload Type	Date	Host	VMs	Mean (%)	SD(%)
Real (PlanetLab)	03/03/2011	800	1052	12.31	17.09
	22/03/2011	800	1516	9.26	12.78
	03/04/2011	800	1463	12.39	16.55
	20/04/2011	800	1033	10.43	15.21
Random	----------	50	50	-------	--------

3.4.2 Workload Data

To make the simulation based evaluation applicable, we evaluate the Markov Prediction Model approach on random workload and real-world publicly available workloads:

- Real Workload (PlanetLab data) [74]: This is provided as a part of the CoMon project; it is a monitoring infrastructure for PlanetLab. In this project, the CPU utilization data is obtained every five minutes from more than a thousand VMs from servers located at more than 500 places around the world. Data is stored in ten different files. We chose two different days from the workload traces gathered during March 2011 and one day from April 2011 of the project. Through the simulations, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day. Table 3-1 shows the characteristics of each workload.
- Random Workload: Requests for provisioning of 50 heterogeneous VMs that fill the full capacity of the simulated data center are submitted by the users. Each VM runs an application with the variable workload, which is modeled to generate the utilization of CPU according to a uniformly distributed random variable. Each application has a length that determines the number of instructions with MI. The application runs for 150,000 MI that is equal to 10 minutes of the execution on 250 MIPS CPU with 100% utilization.

3.4.3 Performance Metrics

To compare the performance of our proposed algorithms with the existing algorithms we have chosen eight metrics which are previously defined: SLA violation, percentage of SLA violation time per active host and SLA%, performance degradation that occurs due to migration of VM from one host to another while balancing load or switching off underutilized servers, average SLA violation which describes how many times allocated resources are less than required resources, total number of VM migration occurred either for hotspot mitigation or for VM consolidation, total energy consumption by the physical resources for executing variable workloads, and finally number of hosts that are switching off.

- SLA Violation: In a cloud environment, SLA is agreed between the service provider and the user to ensure the required level of service. SLA contains various details of service level that will be provided to a user, such as, minimum capacities of CPU, RAM, storage, and bandwidth. In case of SLA violation, a party that is responsible for its breach has to pay a fine to the other party. The CPU usage by a VM arbitrarily varies over time. The host is oversubscribed, i.e. if all the VMs request their maximum allowed CPU performance, and the total CPU demand exceeds the capacity of the CPU. It is defined that when the request for the CPU performance exceeds the available capacity, a violation of the SLA established between the resource provider and the customer occurs. For our studies, SLA violation is calculated as shown in Equation (3.8) [52]:

$$SLA\ Violations\ (SLAV) = SLATAH * PDM \tag{3.8}$$

where SLAV denotes SLA violation, SLATAH represents SLA violation Time per Active Host, and PDM stand for Performance Degradation due to Migrations. Following equations can be used to calculate SLATAH and PDM.

- SLA violation time per active host (SLATAH): is the observation that if a host serving applications is experiencing the 100% utilization, the performance of the applications is bounded by the host's capacity; therefore, VMs are not being provided with the required performance level. In other word, it means SLA violations due to overutilization [52].

$$SLATAH = \frac{1}{J}\sum_{j=1}^{J}\frac{T_{sj}}{T_{aj}} \tag{3.9}$$

where J is number of hosts, T_{sj} is the total time that utilization of host j reaches 100 %, and T_{aj} is the lifetime (total time that host is active) of host j. When host utilization reaches 100 %, the applications performance is bounded by the host.

- Performance degradation due to migration (PDM): Live migration is the process of moving VMs from one host to another one (without suspension), it has a negative impact on user applications performance. Voorsluys et al. [71] show that this impact depends on application behavior, and the performance degradation can be estimated as 10% of CPU utilization. In other word, it means the SLA violations is due to migration.

$$PDM = \frac{1}{V}\sum_{v=1}^{V}\frac{Cd_v}{Cr_v} \tag{3.10}$$

where V is the number of VMs, Cd_v estimated as 10% CPU utilization of VM_v in all migrations, Cr_v is total CPU requested by VM_v.

- Average SLA violation: is measured as the mean of the difference between total requested resources (MIPS) by all the VMs and total allocated resources (MIPS). Equation (3.11) can be used to calculate

$$Average\ SLAV = \frac{\sum_{v=1}^{V}(requestedMIPS) - \sum_{v=1}^{V} allocatedMIPS}{V} \qquad (3.11)$$

where V shows number of VMs

- Overall SLA violation: is measured as the mean of the difference between total requested resources (MIPS) by all the VMs and total allocated resources (MIPS) [68]. Equation (3.12) can be used to calculate

$$Overall\ SLAV = \frac{\sum_{v=1}^{V}(requestedMIPS) - \sum_{v=1}^{V} allocatedMIPS}{\sum_{v=1}^{V}(requestedMIPS)} \qquad (3.12)$$

where V is the number of VMs.

- Number of VM migration: a higher number of VM migrations increases the network load, and results in performance degradation. Equation (3.13) can be used to calculate the number of migrations during a given time interval [22].

$$Migrations(P, t1, t2) = \sum_{j=1}^{J} \int_{t1}^{t2} Mig_j(P, t) \qquad (3.13)$$

where P represents the current placements of VMs, J is the number of hosts, $Mig_j(P, t)$ shows the number of migration of Host j between time intervals $t_1 and\ t_2$ for the placement P.

- Energy Consumption: In order to measure the power consumption of a given host at a time t with placement P [75]. Equation (3.14) can be used to calculate

$$W_j(P,t) = k \ * W_{max} + (1-k) * W_{max} * U_j(P,t) \quad (3.14)$$

where W_{max} is the power consumption of the host at 100% utilization, k is the static power coefficient that is equal to the amount of power consumption by an idle processor. According to [76], an idle processor consumes 70% of the power consumed when its utilization is 100%. Therefore, in our experiments, k is set to 70%. In this model, $U_j(P,t)$ is the current CPU utilization of a host j at time t, which has a linear relationship with the power consumption. Total energy consumption of all the hosts between time t1 and t2, can be calculated using Equation (3.15).

$$Energy(P, t_1, t_2) = \sum_{\mathrm{j}=1}^{J} \int_{t1}^{t2} W_j(\mathrm{P}, t) \quad (3.15)$$

Table 3-2 illustrates the amount of energy consumption of two types of HP G4 and G5 servers at different load levels. The table shows the energy consumption is reduced efficiently when under-utilized PMs switch to the sleep mode [52].

Table 3-2: The energy consumption at different load levels in Watts

Server	sleep	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP G4	10	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP G5	10	93.7	97	101	105	110	116	121	125	129	133	135

- Number of host shutdowns: consolidation is applied to reduce the number of active physical hosts, the quality of VM consolidation is inversely proportional to H, the mean number of active hosts over n time steps [49]:

$$H = \frac{1}{n}\sum_{i=1}^{n} a_i \quad (3.16)$$

where a_i is the number of active hosts at the time $step\ i = 1, 2, \ldots, n$. A lower value of H represents a better quality of VM consolidation.

QoS is mainly affected by the SLA violation, where the SLA violation is affected by number of VMs migration, PDM, and SLATAH metrics.

3.5 Experimental Results

In this section, we first present the impact of the data length of host status history in our algorithm that makes it perform the best on four different VM selection polices with three different PlanetLab workloads and a random workload. We then show the impact of four different VM selection polices on our algorithms. Then, we discuss our experimental results in comparison to the benchmark algorithms. Finally, the impact of proposed placement algorithm on MadMCHD algorithm is investigated.

3.5.1 Maximum Data length of host status history of Markov Model

One of the important parameter for Markov model is to determine the maximum data length. Consequently, we first investigate a different range of data length in order to find the most suitable length for the four different VM selection polices. To perform this experiment, we study this parameter with three different PlanetLab workloads and a random workload. To choose the best data length, we rely on the aforementioned eight metrics. We have observed through this experiment that each data length parameter affects VM selection policies differently. Therefore, we have chosen the data length parameter that performs well in most of four VM selection policies. We have selected a range for data length from 30 to 180. We have not increased the range over 180 because of time complexity.

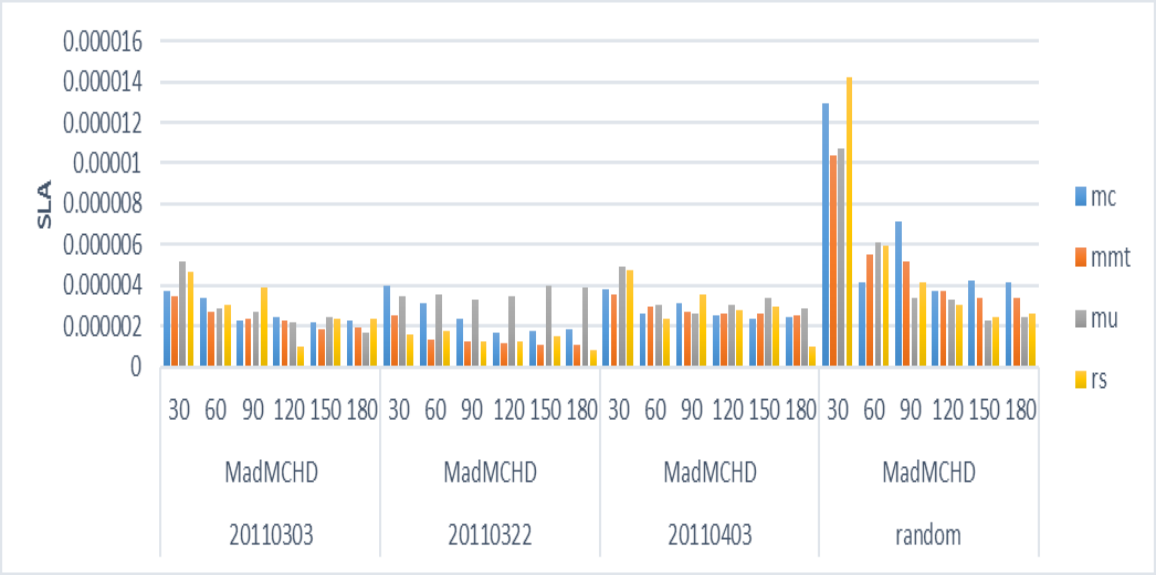

Figure 3-5: The impact of data length on the SLA metric

Figure 3-6: The impact of data length on the number of VM migration metric

We have studied the impact of the mentioned range on the eight metrics. However, for the sake of space, we have shown the impact of data length on SLA violation and number of VM migration metrics as shown in Figure 3-5 and Figure 3-6 respectively. According to these figures, we have chosen the data length parameter, 120, and this parameter is used for the comparison experiments. For instance, we calculate the average

of SLA violation metric when the work load is 20110303 and we have found the following: when the MC policy is used and data length is 30, the average of SLA violation metric is 4.2581. Also, the average SLA violation metric for the data length 60, 90, 120, 150, 180 are 2.9934, 2.78045, 1.984, 2.20512, and 2.0664 respectively. Based on these numbers, we can see that the best data length is 120. From Figure 3-6, when the work load is 20110322 is used, we have found that the average for number of VM migration is 3296 when the data length is 120, while the average of VM migration is 3250 when the data length is 180. Since this is a slight difference, we consider 120 as the most suitable data length to avoid time complexity when the data length is 180.

3.5.2 Comparison with other benchmarks

We are further interested in comparing our proposed algorithms with the state-of-the-art algorithms. To perform this comparison, we employ the aforementioned eight metrics in order to assess our results. Our comparison process is to study the algorithms' performance in the entire selection process which includes host detection, VM selection, and VM placement.

We compare the proposed algorithm, MadMCHD, with the state-of-the-art five host detection algorithms, namely IQR, MAD, LRR, LR, and THR (which is a static threshold set to 0.8) [42, 52, 67]. Besides, we investigate the impact of four well-known VM selection polices on the proposed model, which are described below. The VM selection algorithms include:

- Maximum Correlation (MC) is inspired that high correlation between tasks and resource usage might lead to server overloading. MC uses the multiple correlation coefficient which corresponds to the squared correlation between the predicted and

the actual values of the dependent variable [52].

- Minimum Migration Time (MMT): selects VMs based on the value of the migration time, the less the better. The migration time can be easily computed as the amount of RAM utilized by the VM divided by the additional network bandwidth available for the current allocated host [52].
- Maximum Utilization (MU): Choosing the VMs to migrate from the hotspot based on the largest possible CPU usage can be expected to minimize the number of migrations [41].
- Random Selection (RS): selects the necessary number of VMs by picking them according to a uniformly distributed random variable [40].

For IQR, LR, LRR, MAD, THR, and MadMCHD, we use the well-known placement method which is called PABFD [42,52]. The main goal of these experiments is to substantiate the threshold adaptability in hypothesis by evaluating the performance of the proposed algorithm across single workload (20110322) that traces from more than a thousand PlanetLab servers and one random workload. In the following we compare our results with the minimum value for each selection algorithms when applied to host detection algorithms. For example, when selection algorithm MC is applied to all state-of-the-art detection algorithms, we compare our result with the one which gives minimum value (example SLA % in Figure 3-7).

From the simulation results depicted in Figure 3-7 and Figure 3-8, it is completely obvious that the proposed algorithm significantly outperforms the other algorithms in terms of SLA violation for both 20110322 PlanetLab real workload and the random workload, since our proposed host load detection algorithm avoids immediate VMs

migration. It reduces SLA violation metric by 97.19%, 96.16%, 92.34%, and 90% for the real workload, and by 98.25%, 97.98%, 98.39, and 98.54% for the random workload for VM selection policies MC, MMT, MU, and RS respectively.

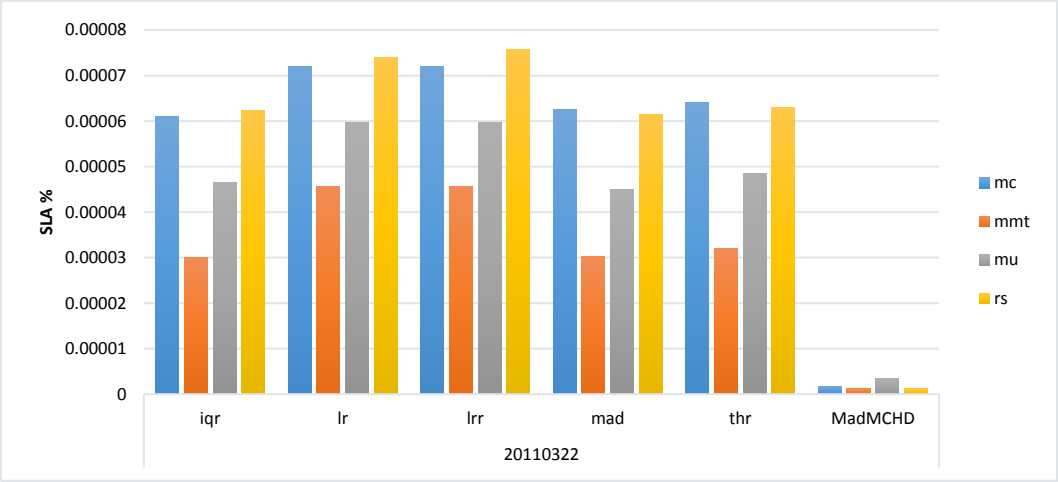

Figure 3-7: SLA violation for real workload trace

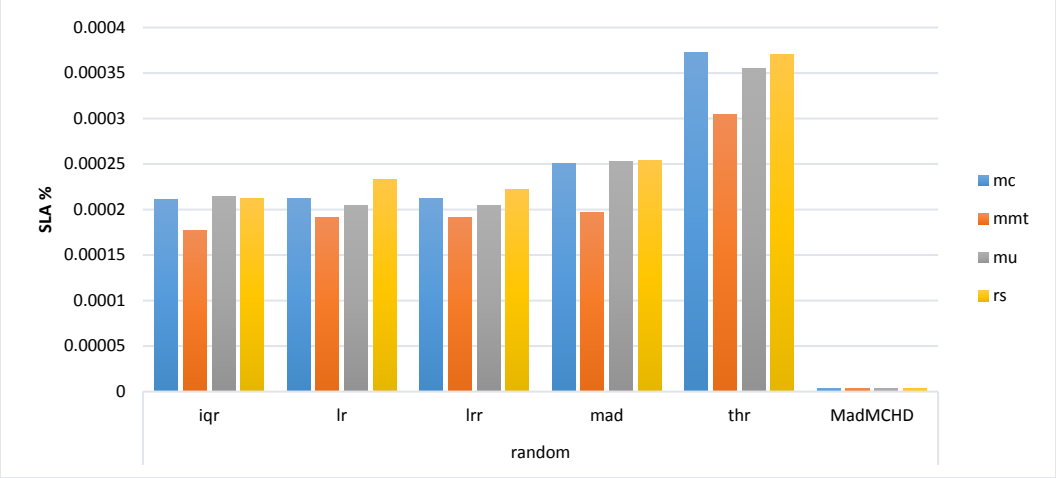

Figure 3-8: SLA violation for a random workload trace

From the simulation results depicted in Figure 3-9 and Figure 3-10, it is completely obvious that the proposed algorithm significantly outperforms the other algorithms in terms of number of VM migrations for both 20110322 PlanetLab real workload and the random workload, since our proposed algorithm avoids immediate VMs migration. The

proposed host load detection algorithm reduces number of VM migrations metric by 88.73%, 89.90%, 85.35%, and 89.15% for the real workload, and by 83.97%, 87.74%, 84.61%, and 80.07% for the random workload for VM selection policies MC, MMT, MU, and RS respectively.

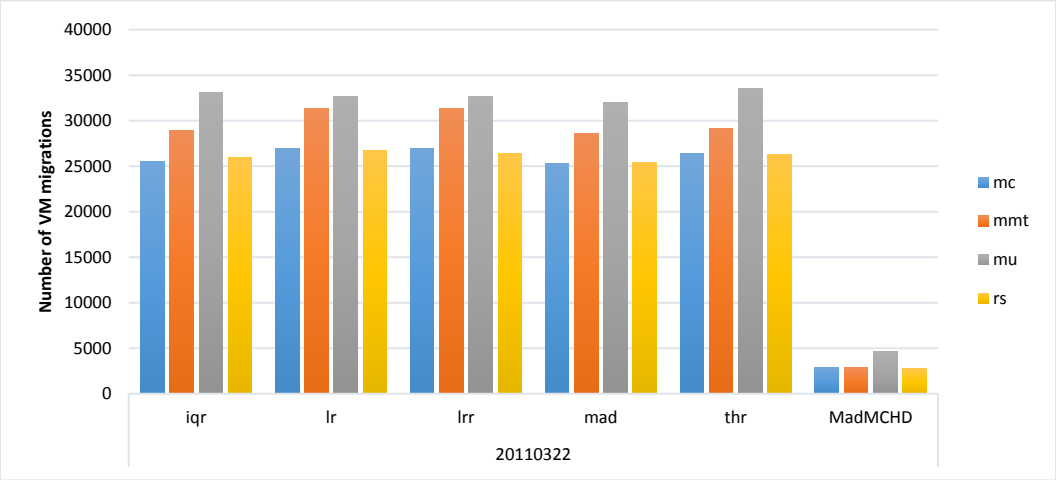

Figure 3-9: Number of VM migrations for real workload trace

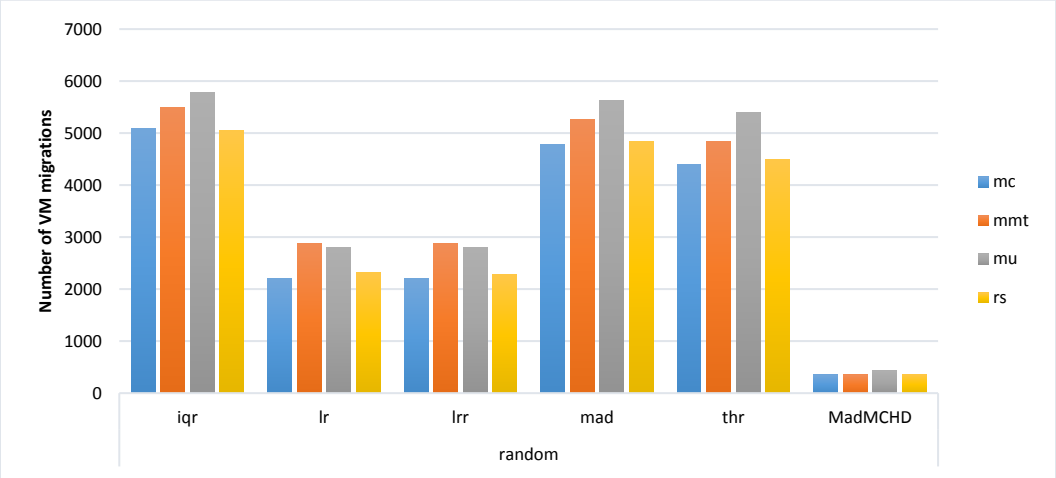

Figure 3-10: Number of VM migrations for a random workload trace

From the simulation results depicted in Figure 3-11 and Figure 3-12 it is completely obvious that the proposed algorithm significantly outperforms the other algorithms in terms of PDM for both 20110322 PlanetLab real workload and the random

workload, since our proposed algorithm reduces total CPU requested by VMs. The proposed host load detection algorithm reduces PDM migration metric by 71.02%, 72.11%, 58.52%, and 79.05% for the real workload, and by 78.28%, 73.87%, 83.35%, and 83.56% for the random workload for VM selection policies MC, MMT, MU, and RS respectively.

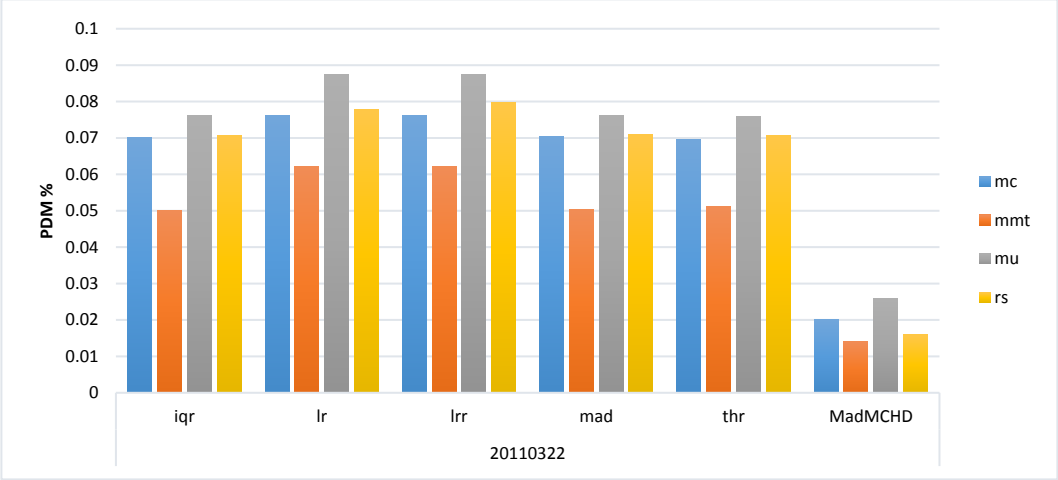

Figure 3-11: Performance degradation for real workload trace

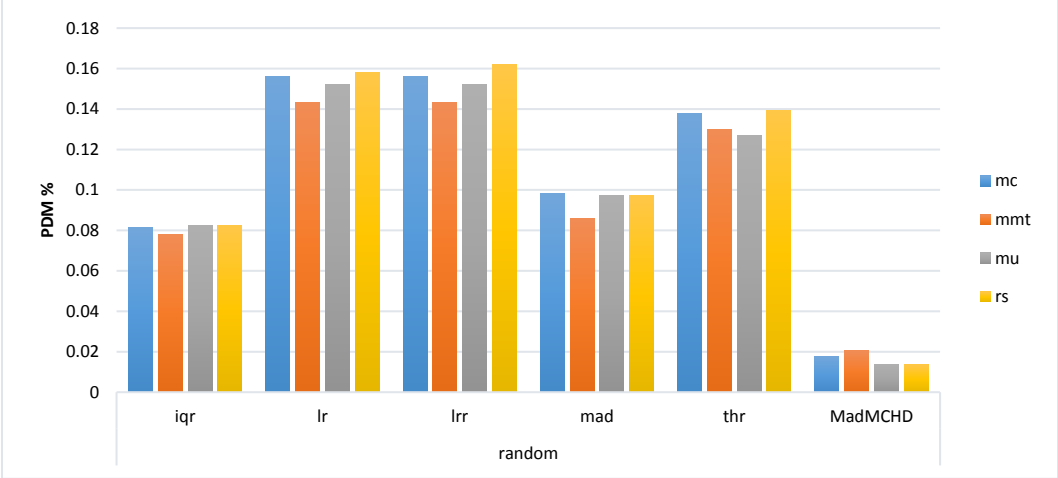

Figure 3-12: Performance degradation for a random workload trace

From the simulation results depicted in Figure 3-13 and Figure 3-14, it is completely obvious that the proposed algorithm significantly outperforms the other

algorithms in terms of SLATAH for both 20110322 PlanetLab real workload and the random workload, since our proposed algorithm reduces total time of staying overutilized. The proposed host load detection algorithm reduces SLATAH migrations metric by 90.27%, 86.29%, 78.31%, and 90.83% for the real workload and by 84.58%, 86.30%, 82.19%, and 83.40% for the random workload for VM selection policies MC, MMT, MU, and RS respectively.

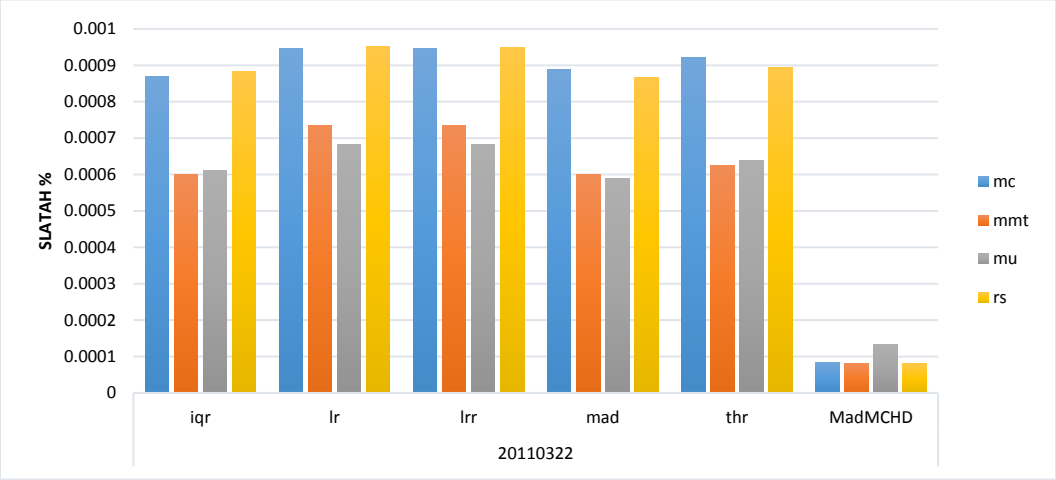

Figure 3-13: SLA violation time per active host for real workload trace

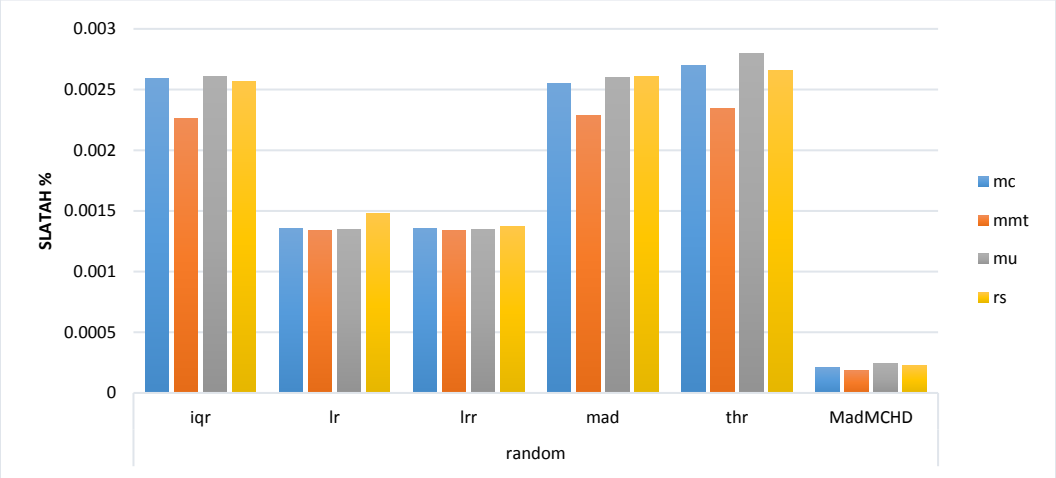

Figure 3-14: SLA violation time per active host for a random workload trace

Figure 3-15 shows that the proposed algorithm slightly outperforms the other algorithms in terms of the average SLA violation for 20110322 PlanetLab real workload. Figure 3-16 shows that proposed algorithm is almost similar to the MAD and IQR algorithms in term of the average SLA violation, and the performance of the proposed algorithm is not much better than that of LR, LRR and THR algorithms for the random workload.

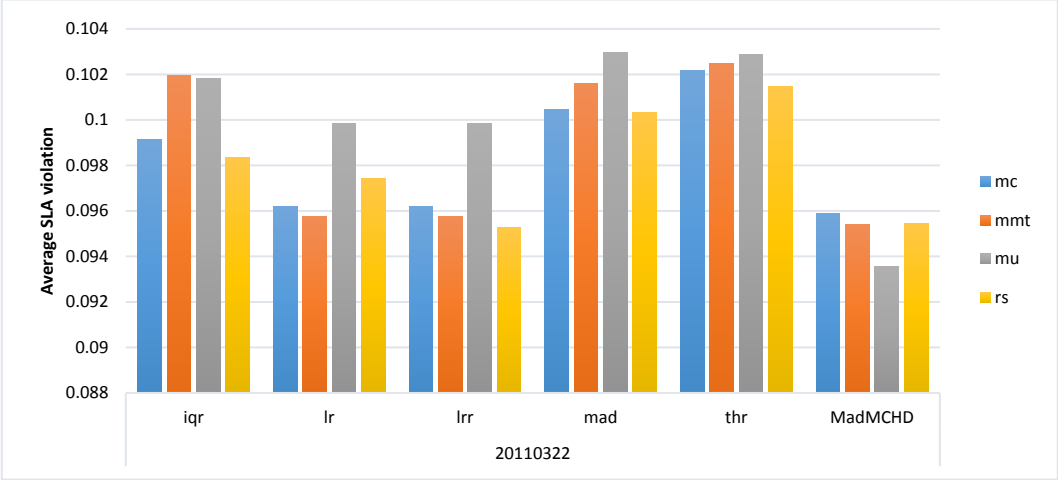

Figure 3-15: average SLA violation for real workload trace

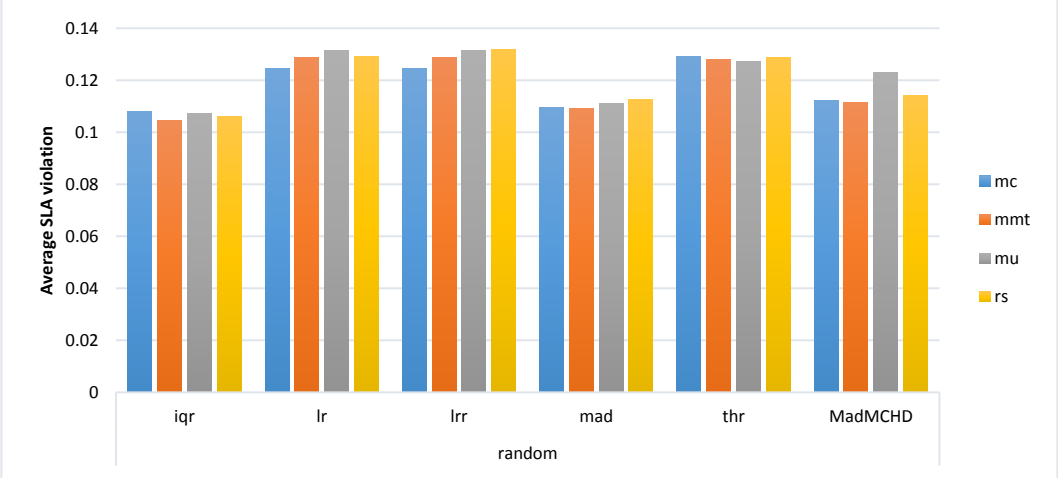

Figure 3-16: average SLA violation for a random workload trace

Figure 3-17 shows that the proposed algorithm slightly outperforms the lr and lrr algorithms in term of overall SLA violation for 20110322 PlanetLab real workload. It should be noted that the performance of THR, MAD and IQR algorithms still outperform the other algorithms. It is completely obvious from Figure 3-18 that the proposed algorithm significantly outperforms the other algorithms in terms of overall SLA violation for the random workload. The proposed host load detection algorithm reduces overall SLA violation metric by 81.05%, 81.22%, 76%, and 80.07% for VM selection policies MC, MMT, MU, and RS respectively.

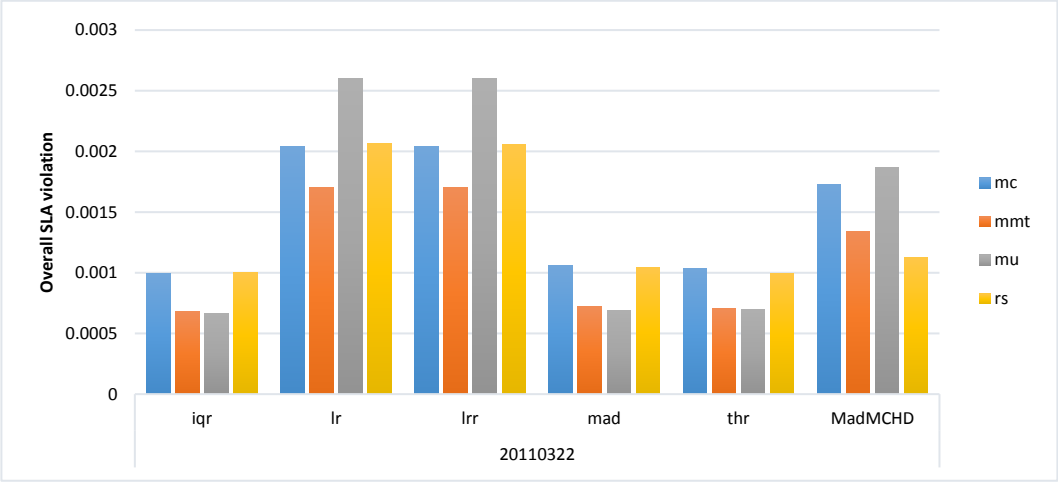

Figure 3-17: overall SLA violation for real workload trace

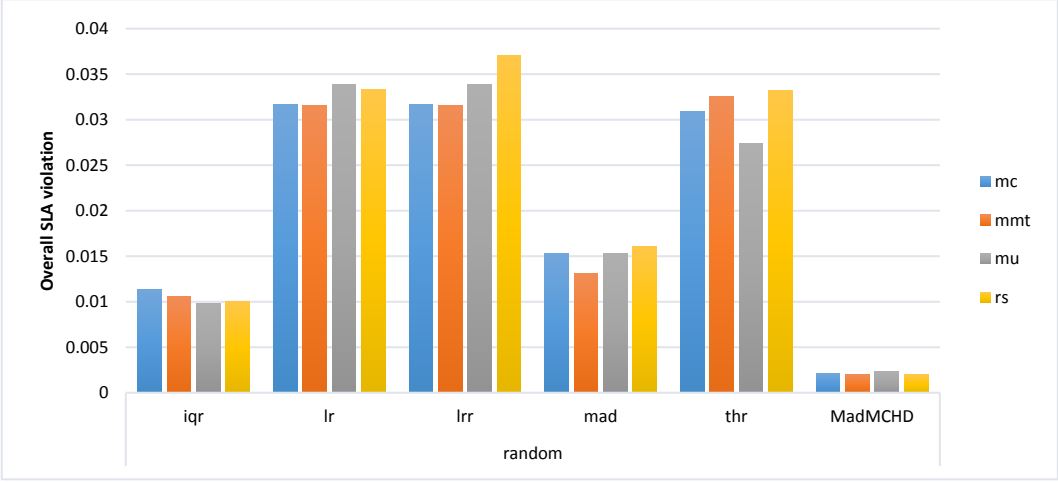

Figure 3-18: overall SLA violation for a random workload trace

Figure 3-19 and Figure 3-20 show that the proposed algorithm is almost similar to the THR, MAD and IQR algorithms in term of the energy consumption. It should be noted that the performance of the proposed algorithm is not much worse than that of LR and LRR algorithms.

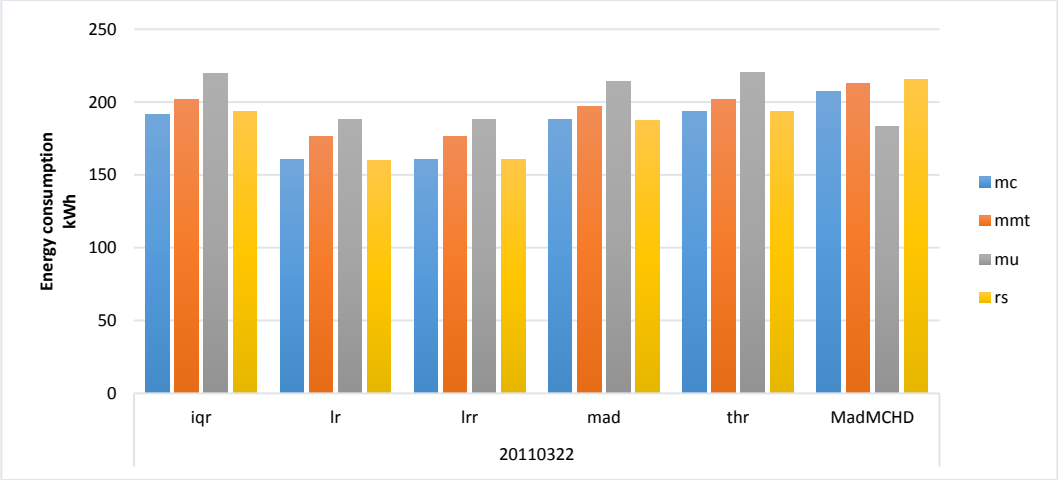

Figure 3-19: energy consumption for real workload trace

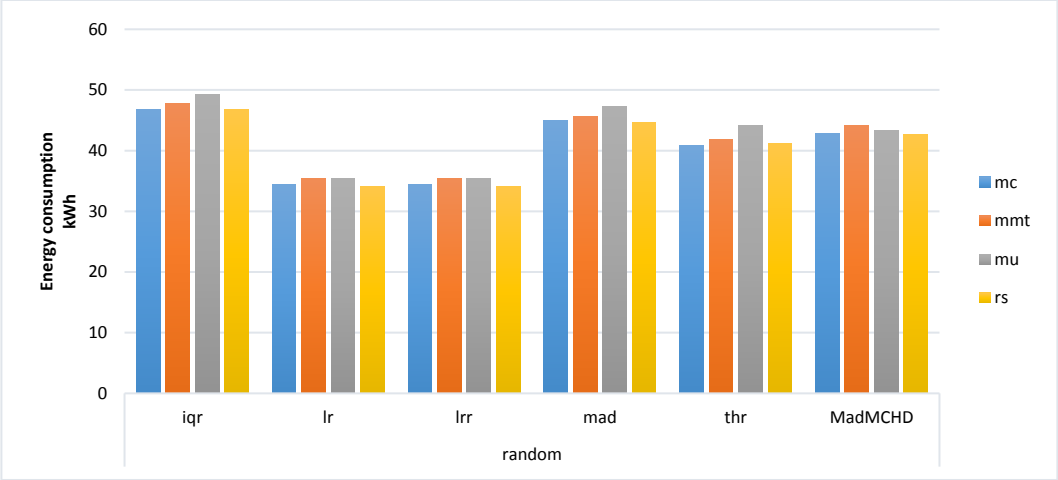

Figure 3-20: energy consumption for a random workload trace

From the simulation results depicted in Figure 3-21 and Figure 3-22, it is completely obvious that the proposed algorithm significantly outperforms the other algorithms in terms of number of host shutdowns for both 20110322 PlanetLab real

workload and the random workload, since our proposed algorithm reduces number of active hosts. The proposed host load detection algorithm reduces number of host shutdowns metric with minimum improvement reach by 82.44%, 85.44%, 80.31%, and 82.52% for the real workload, and by 81.45%, 84.36%, 82.59%, and 81.42% for the random workload for VM selection policies MC, MMT, MU, and RS respectively.

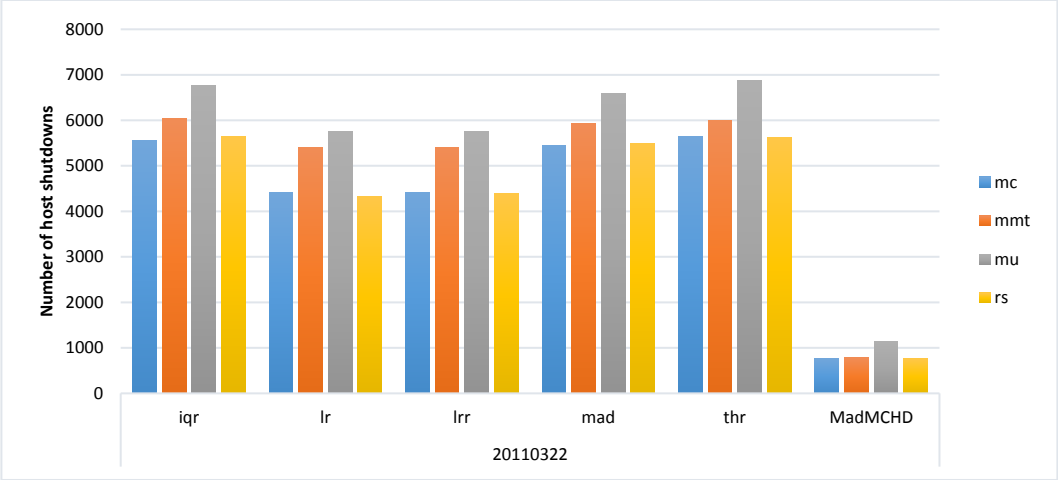

Figure 3-21: number of host shutdowns for real workload trace

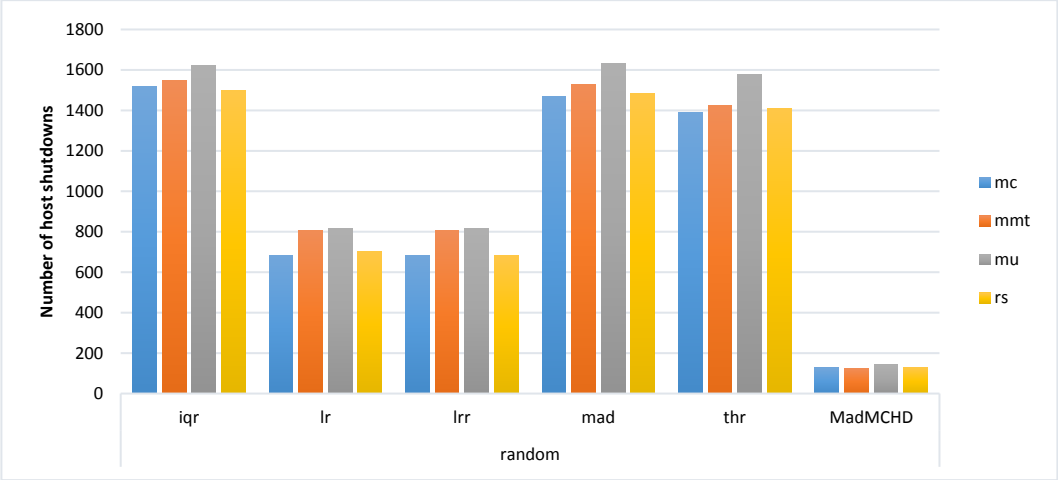

Figure 3-22: number of host shutdowns for a random workload trace

Why a lower number of host shutdowns is better with similar energy consumption? Host shutdown due to improper VM migration in case of underloaded hosts may result in a host not to stay in the shutdown mode for a long time which does not result in a real improvement in the power consumption. In our case a significant decrease in number of host shutdown indicates a better scheduling of VMs in the data center. At the same time there is an improvement in the SLA violation, a reduction in the number of VM migrations, and an increase in the resource utilization.

QoS is satisfied by reducing the number of VM migrations, and the percentage of PDM and SLATAH metrics, which in turn has an effect on reducing the percentage of SLA violations.

3.6 The Impact of proposed placement algorithm on MadMCHD algorithm.

We investigate the impact of our proposed MPABFD placement algorithm when it is used in combination with our proposed MadMCHD host detection algorithm, termed as MadMCHDPP as compared to another combination where the host detection algorithm MadMCHD is used with the state-of-the-art placement algorithm PABFD, termed as MadMCHD. For both combinations, the four selection policies are used, which are mc, mmt, mu and rs. Figure 3-23 shows that the proposed combination MadMCHDPP reduces overall SLA violation metric by 47.80%, 45.52%, 47.03% and 14.86% for the real workload for VM selection policies MC, MMT, MU, and RS respectively. On the other hand, the proposed combination MadMCHDPP is almost similar to MadMCHD in the other metrics.

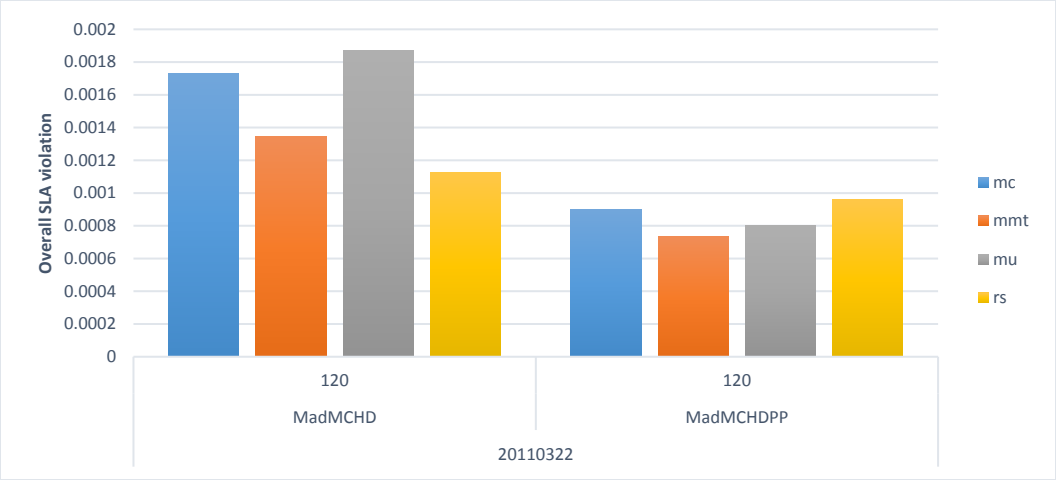

Figure 3-23: overall SLA violation for a random workload trace

3.7 Summary

In this chapter, we present Median Absolute Deviation Markov Chain Host Detection algorithm (MadMCHD) based on a dynamic utilization threshold. The proposed algorithm avoids immediate VMs migration in cloud data center by predicting the future host CPU utilization. The current host CPU utilization is calculated and compared with the lower and the upper threshold to determine the current host state. The future host state is predicted using the proposed Markov host prediction model. The proposed algorithm determines when to migrate VMs to achieve server consolidation and load balancing for all the host states.

We present Markov Power Aware Best Fit Decreasing (MPABFD) algorithm to enhance VMs placement process. The future candidate host load state is predicted to avoid overloaded state of that host after a short period. We combine the proposed algorithms in the selection process phases in the live migration for better performance, MadMCHD as a host detection algorithm, MPABFD as a VM placement algorithm, and

some of the state of the art algorithms as a VM selection. We investigate the impact of these VM selection polices on the proposed model.

The experimental results show that increasing of the data length of Markov model results in an enhanced performance until a certain value, after which not much improvement in performance is obtained. This value is chosen to not further increase the time complexity of the system.

The experimental results show that MadMCHD algorithm can minimize SLA violation rate, number of VM migration, and the other metrics significantly as compared to the most commonly used THR, MAD, IQR, LR and LRR algorithms. The new combination of the proposed MadMCHD and MPABFD algorithms shows overall SLA violation is reduced significantly.

Chapter 4

Minimizing Biased VM Selection

4.1 Overview

VM selection algorithm selects one or more VMs from the full set of VMs running on a given overload host, once a decision to migrate VMs from that host is made to achieve host/server consolidation and load balancing in cloud data centers while satisfying the QoS constraints. Presently, VM selection is a crucial decision for resource management in the cloud data center management, especially with high dynamic environment. In this Chapter, two new VM selection algorithms are proposed, namely Minimum VM Migrated Count and Minimum migration time Minimum VM Migrated Count to avoid frequent SLA violation on the same VM. New metrics are proposed to compare with other VM selection algorithms. Our proposed algorithms are evaluated through CloudSim simulation on different types of PlanetLab real and random workloads. The experimental results demonstrate that the proposed algorithms show significant reduction in the Maximum number of VM migrated count and the degree of load balancing of VMs migrated count with the other state of the art algorithms.

4.2 Proposed VM Selection Policies

The process of migration not only makes the VM unavailable for a certain amount of time but also consumes the network and CPU resources from both source and destination hosts. This study proposes VM selection policies that resolve biased VM selection in live VM migration, resulting in a fair SLA violation on all the VMs while keeping the same percentage in the other metrics.

- Minimum VM Migrated Count (MiMc): The algorithm selects the VM to migrate from the host overloaded based on the minimum number of VM migrated count.
- Minimum Migration Time Minimum VM Migrated Count (MmtMiMc): The algorithm first selects VMs with the minimum amount of RAM to minimize the live migration time [41] and sorts them in increasing order. Then, out of the selected subset of VMs, the algorithm selects the VM with the minimum number of VM migrated count.

Algorithm 4- 1: Minimum VM Migrated Count (MiMc) algorithm

```
1   Input: OverloadedHost.
2   Output: a VM to migrate.
3   min_migrated_count ← Max
4   selected_vm ← None
5   vmList ← OverloadedHost.getVmList()
6   foreach vm in vmList do
7       migrated_count = vm.getMigrated_count
8       if migrated_count < min_migrated_coun then
9           min_migrated_count ← migrated_count
10          selected_vm ← vm
11  return selected_vm
```

Algorithm 4-2: Minimum Migration Time Minimum VM Migrated Count (MmtMiMc) algorithm

```
1   Input: OverloadedHost, vms_ram_values.
2   Output: a VM to migrate.
3   min_migrated_count ← Max
4   selected_vm ← None
5   vmList ← OverloadedHost.getVmList()
6   vmList.sortDecreasing_vms_ram_values()
7   For (int i = 0; i < 4; i++)
8   vmList2 [i] ← vmList[i]
9   foreach vm in vmList2 do
10      migrated_count = vm.getMigrated_count
11      if migrated_count < min_migrated_coun then
12          min_migrated_count ← migrated_count
13          selected_vm ← vm
14  return selected_vm
```

We compare proposed algorithms, MiMc and MmtMiMc, with three state-of-the-art VM selection algorithms, namely MC, MMT, and MU [41, 52]. Besides, we investigate the impact of the four well-known host detection policies on the proposed algorithm. These VM host detection algorithms include:

- Averaging threshold-based algorithm (THR) computes the mean of the n last CPU utilization values and compares it to the previously defined threshold. The algorithm detects underload state if the average of the n last CPU utilization measurements is lower than the specified threshold.
- Median Absolute Deviation (MAD) specifies a lower threshold empirically, while the upper threshold is calculated using the median of the absolute deviation from the medians of the CPU usage data sets.
- InterQuartile Range (IQR) is another approach to determine the upper threshold, while the lower threshold is determined empirically as before.
- Local Robust Regression (LRR) compares the maximum migration time to an

expected value and weights it before deciding of overloading in the host.

We used the same VM placement method which is called PABFD [42, 52]. The VM allocation algorithm selects the destination host to receive the migrated VM, which causes the least increase in the power consumption. The algorithm relies on the traditional greedy algorithm to optimize the allocation of VMs.

4.3 System Model

The target system is an IaaS environment, represented by a large-scale data center. The data center consists of a maximum of J heterogeneous hosts where each host contains multiple VMs. Multiple VMs can be allocated to each host through VMM. Besides, each host and VM are characterized by the CPU performance metrics defined in terms of MIPS, the amount of RAM and network bandwidth. The target system model is depicted in Figure 4-1 [42].

As shown in the Figure 4-1, the system model consists of global and local manager. Users submit their needs for provisioning of M heterogeneous VMs. The local managers, which are part of VMM, resides on each node and are responsible for keeping continuous monitoring of a node's CPU utilization, resizing the VM in accordance with their resource needs and making decision about when and which VMs have to be migrated from the node. The global manager resides on a master node and gathers information from the local managers to keep the check of the general view of the utilization of resources. The global manager gives commands for the optimization of the VM placement. VMMs do actual resizing, migration of VMs and changes in power states of the nodes.

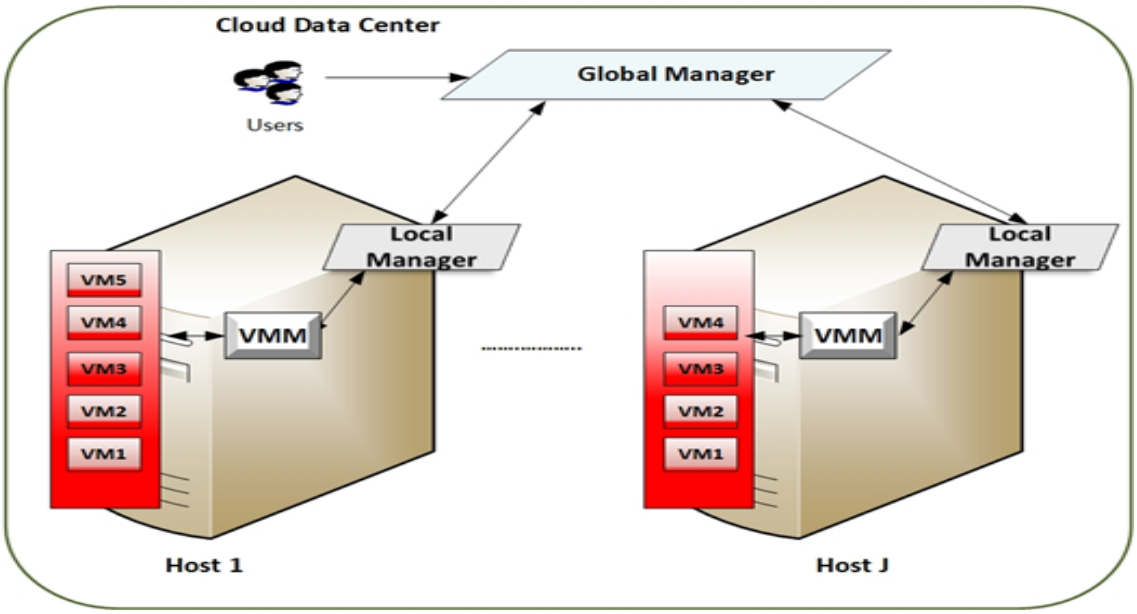

Figure 4-1: System Model

4.4 Experimental Setup

To evaluate the efficiency of our algorithms with the existing algorithm, we have used the same experiment setup as used in [42] with some different workload as explained in section 3.4.1 and section 3.4.2 of Chapter 3.

To make the simulation based evaluation applicable, we evaluate the proposed VM selection approaches on random workload and three real-world [42] publicly available.

To compare the performance of our proposed algorithms with the existing algorithms we have considered five metrics. Three of them are previously defined in the literature, which are SLA violation, total energy consumption by the physical resources for executing variable workloads, and total number of VM migrations occurred either for hotspot mitigation or for VM consolidation. These three metrices are explained in Section 3.6 of Chapter 3. We propose two new metrics, which are the maximum number of VM

migrated count and the degree of load balancing of VMs migrated count, and are precisely defined below:

- Maximum number of VM migrated count: higher number of VM migrated count increases violation on the VM, and results in performance degradation. Following equation can be used to calculate the Maximum number of VM migrated count during a given time interval.

$$migrated\ count(P, t1, t2) = Max(\int_{t1}^{t2} Mig_{VM1}(P,t) , \int_{t1}^{t2} Mig_{VM2}(P,t) , \dots , \int_{t1}^{t2} Mig_{VMn}(P,t)) \quad (4.1)$$

where P represents the current placements of VM, $Mig_{VMn}(P,t)$ shows the number of migration of VM n between time intervals $t_1 and\ t_2$ for the placement P.

- Degree of load balancing of VMs migrated count: a lower number of degree of load balancing reduces biased selection among VMs, resulting in a fair SLA violation on all the VMs. Degree of load balancing is calculated by the variance of the VMs migrated count as indicated in the following equation:

$$Degree\ of\ load\ balancing = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (m_i - \bar{m})^2} \quad (4.2)$$

$$\bar{m} = \frac{1}{N} * \sum_{i=1}^{N} m_i \quad (4.3)$$

where m_i represents migrated count of VM_i, N is the number of VMs, and $\bar{m}$ is average VM migrated count as calculated using Equation (4.3).

4.5 Experimental Results

We selected five performance metrics to compare the proposed algorithms with the existing algorithms, which are SLA violation, total energy consumption, the total number of VM migrations, and the newly proposed Maximum number of VM migrated count and Degree of load balancing of VMs migrated count. We compare with VM selection algorithms presented in [40, 52, 67] including MC, MMT, and MU among four well-known host detection algorithms in [52, 67, 42] including IQR, LRR, MAD, and THR. The main goal of these experiments is to substantiate the threshold adaptability in hypothesis by evaluating the performance of the proposed algorithm across four workloads. The four workloads include three real workloads (20110303, 20110322 and 20110403) that traces from more than a thousand PlanetLab servers and one random workload.

From the simulation results depicted in Figure 4-2 and Figure 4-3, it is completely obvious that the proposed algorithms significantly outperform the other algorithms in terms of Maximum number of VM migrated count and Degree of load balancing of VMs migrated count for all the real workload traces among four host detection policies.

Figure 4-2 shows that MiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 41.03%, 68.92%, and 66.19% as compared to VM selection policies MC, MMT, and MU respectively when the work load is 20110303. There is almost the same reduction in the 20110322 and 20110403 workloads. Figure 4-2 also shows that MmtMiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 12.50%, 52.03%, and 52.52% as compared to VM selection

policies MC, MMT, and MU respectively when the work load is 20110303. There is almost the same reduction in the 20110322 and 20110403 workloads.

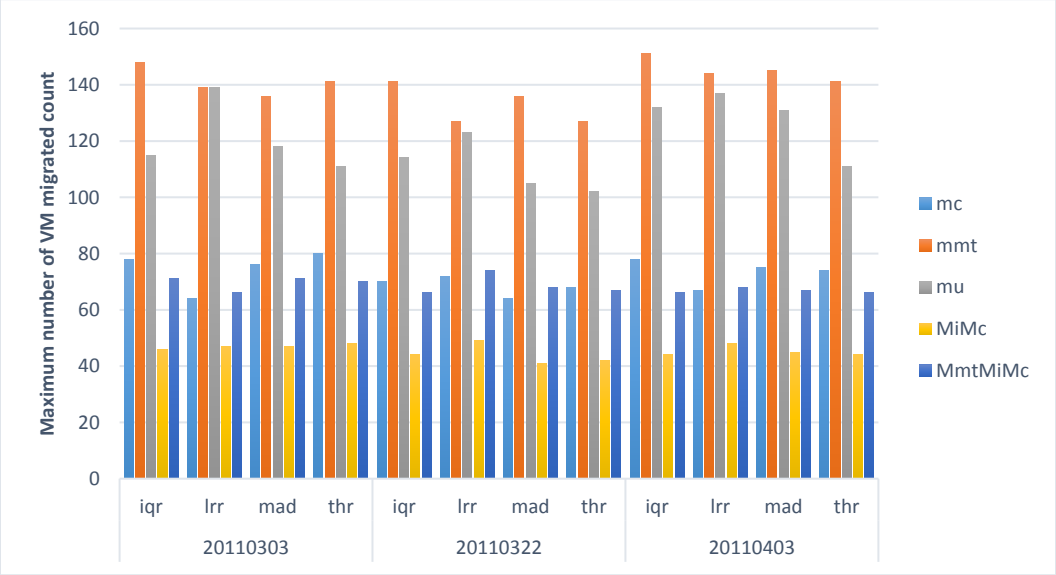

Figure 4-2: Maximum number of VM migrated count for real workload traces

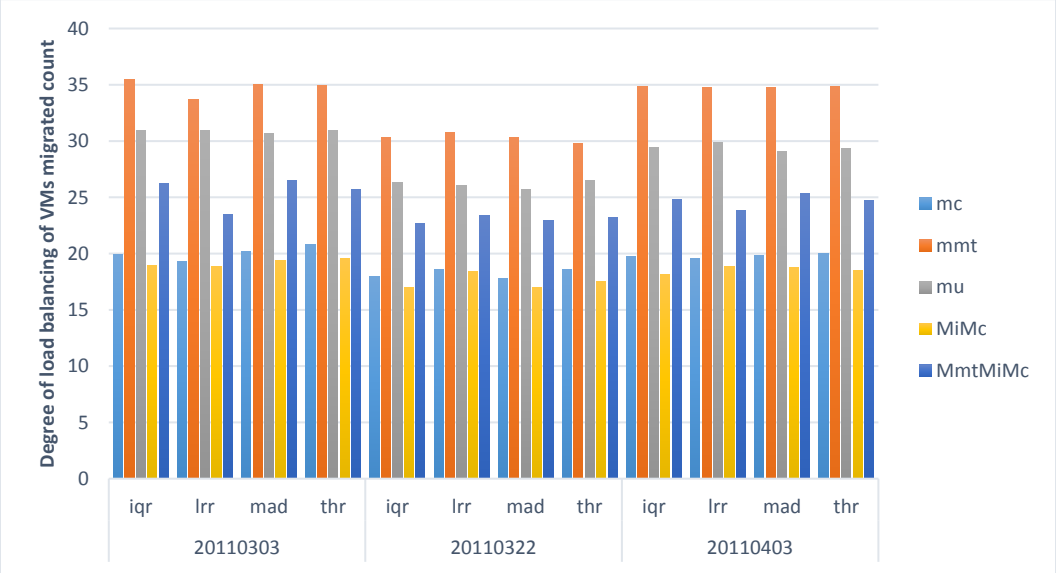

Figure 4-3: Degree of load balancing of VMs migrated count for real workload traces

Figure 4-3 shows that the Degree of load balancing of VMs migrated count metric is reduced up to 4.63%, 46.48%, and 38.62% for MiMc as compared to the VM selection policies MC, MMT, and MU respectively when the work load is 20110303. It should also

be noted that almost the same reduction is obtained in the 20110322 and 20110403 workloads. Figure 4-3 also shows that for the proposed MmtMiMc algorithm, the Degree of load balancing of VMs migrated count metric is reduced up to 25.85% and14.96% for VM selection policies MMT, and MU respectively when the work load is 20110303. It should also be noted that almost the same reduction is obtained in the 20110322 and 20110403 workloads.

Figure 4-4 compares the SLA violation of the two proposed algorithms with the ones in the literature. The proposed MmtMiMc algorithm reduces SLA violation up to 32.14%, 25.85%, and 14.96% for MC, MU, and MiMc respectively when the work load is 20110303. But MMT algorithm still outperforms as the best among the others.

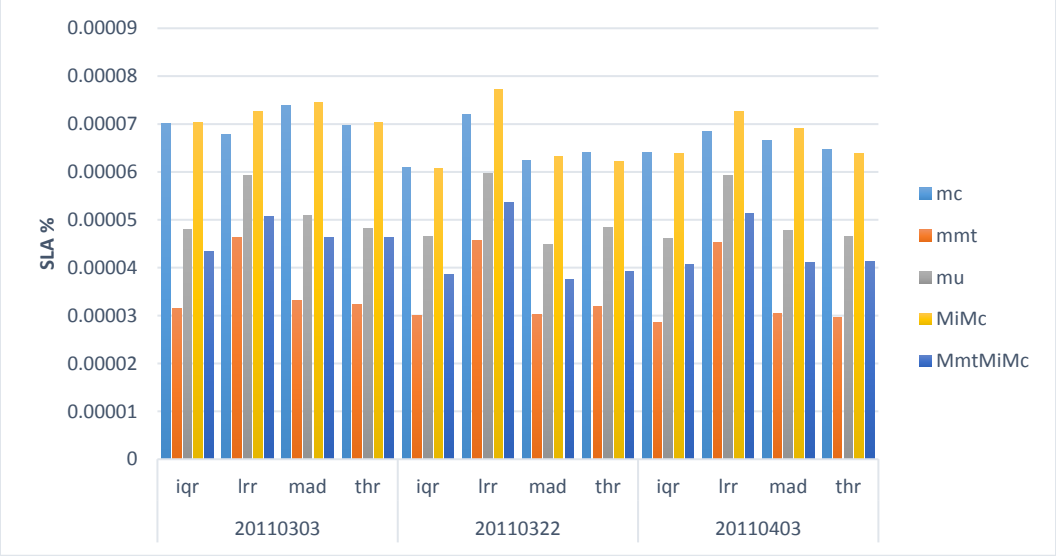

Figure 4-4: SLA violation for real workload traces

Figure 4-5 shows that the proposed algorithms MiMc and MmtMiMc outperform mmt and mu, and are similar to the mc algorithm in terms of number of VM migration. The proposed MiMc and MmtMiMc VM selection algorithms reduce number of VM migrations metric up to 14.42%, and 20.91% as compared to the VM selection policies

MMT, and MU respectively when the work load is 20110303. There is almost the same reduction in the 20110322 and 20110403 workloads.

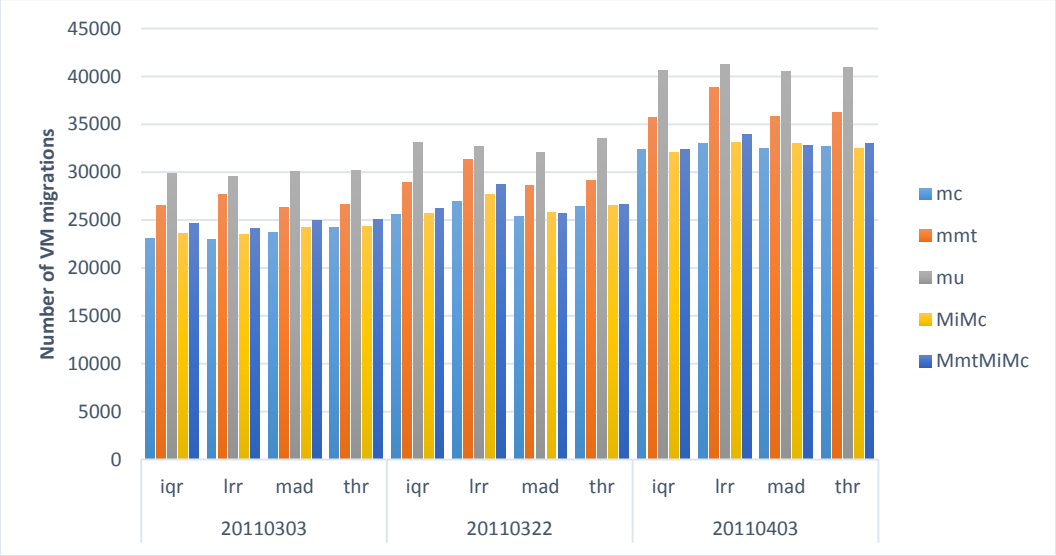

Figure 4-5: Number of VM migrations for real workload traces

It can be seen from Figure 4-6 that the proposed algorithms are slightly better than MMT, and MU and similar to MC in terms of the energy consumption.

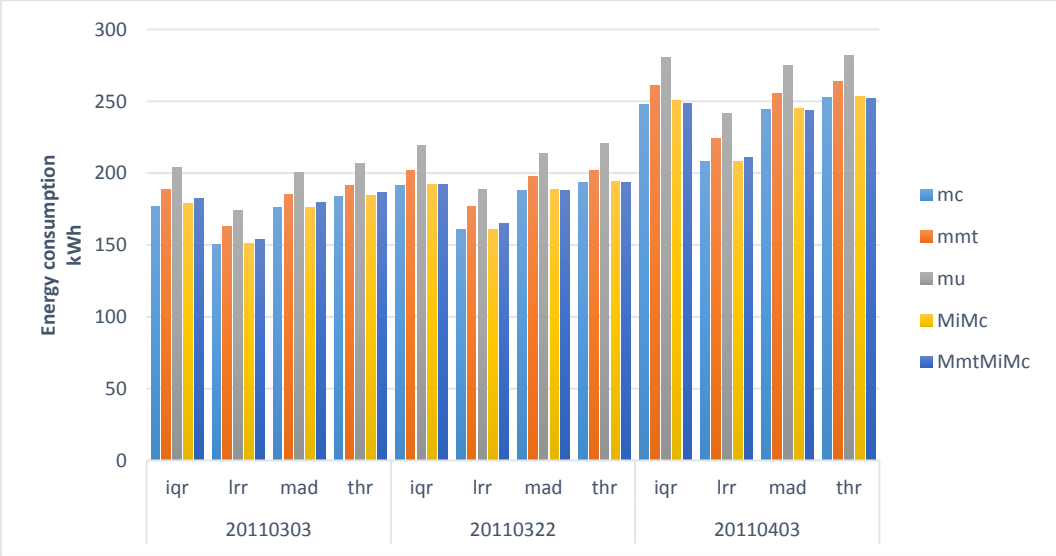

Figure 4-6: Energy consumption for real workload traces

From the simulation results depicted in Figure 4-7 and Figure 4-8, it is completely obvious that the proposed algorithms significantly outperform the other algorithms in terms of Maximum number of VM migrated count and Degree of load balancing of VMs migrated count for the random workload. The proposed MiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 25.41%, 52.11%, and 38.75% as compared to VM selection policies MC, MMT, and MU, respectively. It reduces Degree of load balancing of VMs migrated count metric up to 89.74%, 97.44%, and 90.67% as compared to VM selection policies MC, MMT, and MU, respectively.

Figure 4-7 and Figure 4-8 show that MmtMiMc VM selection algorithm reduces Maximum number of VM migrated count metric up to 23.77%, 57.50%, and 36.25% as compared to VM selection policies MC, MMT, and MU, respectively. It reduces Degree of load balancing of VMs migrated count metric up to 87.68%, 96.91%, and 88.78% as compared to VM selection policies MC, MMT, and MU, respectively.

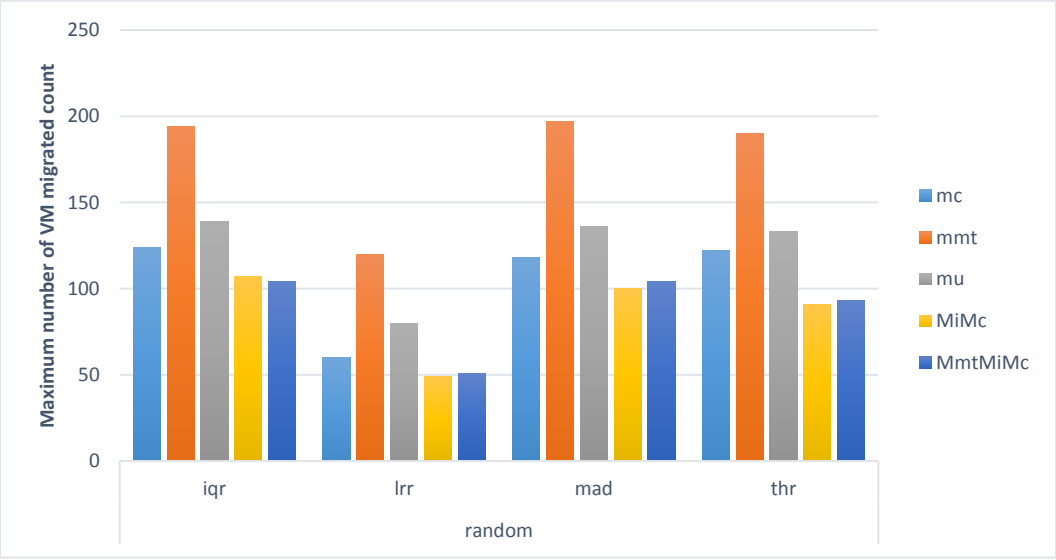

Figure 4-7: Maximum number of VM migrated count for a random workload trace

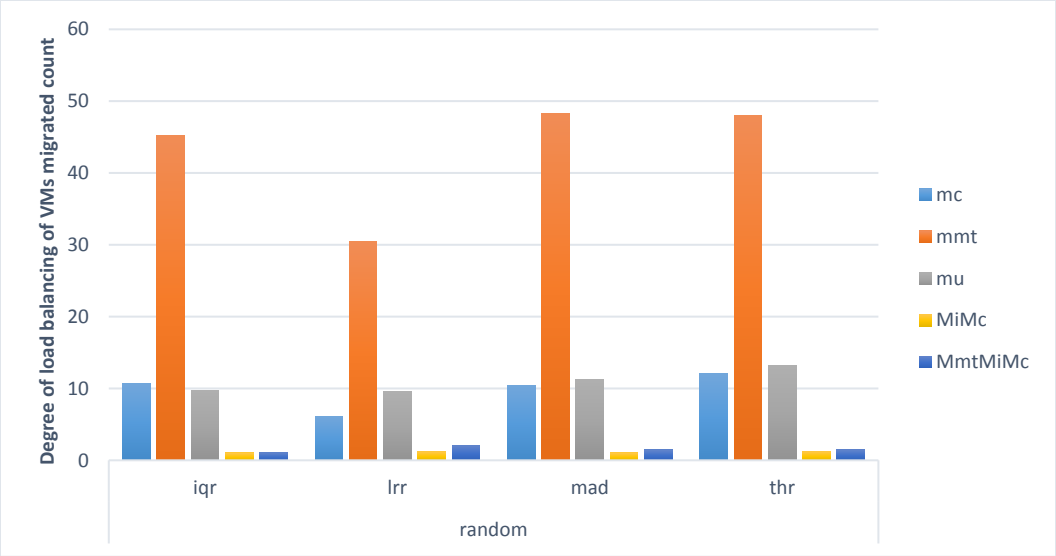

Figure 4-8: Degree of load balancing of VMs migrated count for a random workload trace

It can be seen from Figure 4-9 that the proposed algorithms are similar to the MC, and MU in terms of SLA violation. It should be noted that the performance of the proposed algorithms is not much worse than that of MMT algorithm.

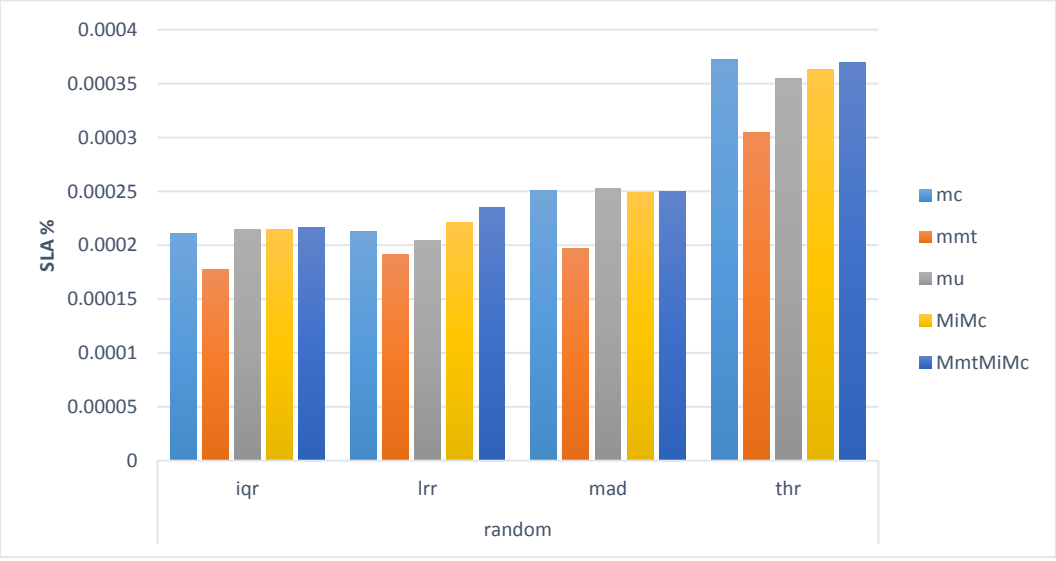

Figure 4-9: SLA violation for a random workload trace

Figure 4-10 shows that the proposed algorithms MiMc and MmtMiMc outperform MMT and MU, and are similar to the MC algorithm in terms of number of VM

migration. MiMc and MmtMiMc algorithms reduce number of VM migrations metric up to 8.35%, and 17.93% for VM selection policies: MMT, and MU respectively.

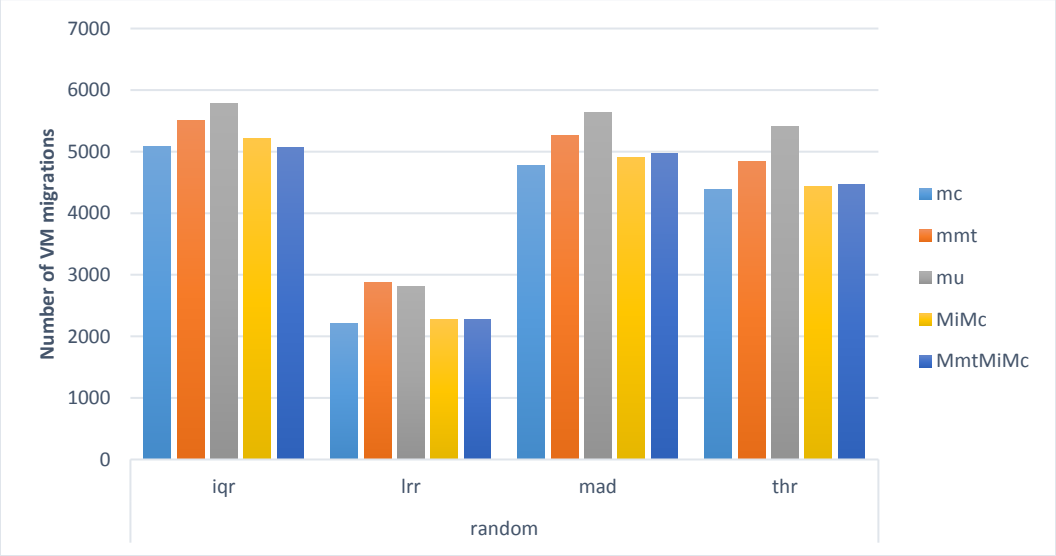

Figure 4-10: Number of VM migrations for a random workload trace

It can be seen from Figure 4-11 that the proposed algorithms are similar to the MC, MMT and MU in terms of the energy consumption for a random workload trace.

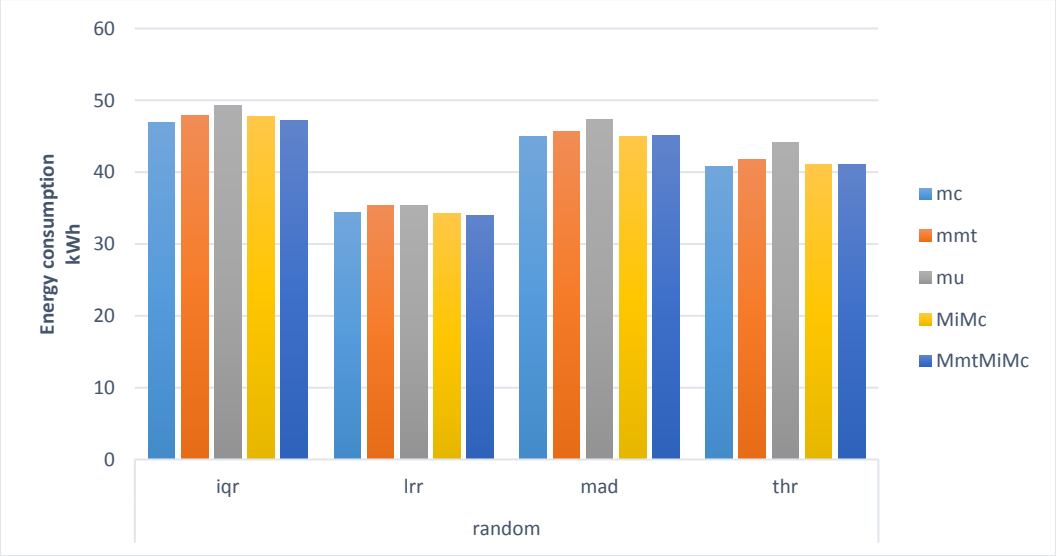

Figure 4-11: Energy consumption for a random workload trace

It should be noted that the improvements in both Maximum number of VM migrated count and Degree of load balancing of VMs migrated count have direct impact on the performance degradation for each VM in the cloud data center. These improvements have impact on avoiding VM migration frequently resulting in a fair SLA violation on all the VMs.

4.6 Summary

In this chapter, we present Minimum VM Migrated Count (MiMc) and Minimum migration time Minimum VM Migrated Count (MmtMiMc) algorithms that resolve biased VM selection in live VM migration. The proposed algorithms avoid frequent SLA violation on the same VM in cloud data center by selecting the VM to migrate from the overloaded host based on VM migrated count. The proposed algorithms determine which VMs will be selected to migrate from the overloaded host to underloaded host to achieve server consolidation and load balancing. The experimental results show that the proposed algorithms can minimize maximum number of VM migrated count and degree of load balancing of VMs migrated count significantly compared to the most commonly used MC, MMT and MU algorithms, resulting in a fair SLA violation on all the VMs, while keeping the same percentage in the other defined metrics.

Chapter 5

Proactive Selection Process for VM Migration Across Cloud Data Centers

5.1 Overview

Live VM migration is a technique that migrates a VM and its application from one host to another in the same data center, which is called LAN migration, or in a different data center, which is called WAN migration. Live VM migration across cloud data centers are useful for several cases despite the costs related to storage migrations and the overheads of network reconfiguration, such as maintenance and upgrades, and large data centers having computing infrastructure around the world that migrate VMs to follow the sun without affecting the end user experience. In this chapter, we propose a new VM selection algorithm, namely Minimum Migration Time Maximum User Ratio to be a proactive solution for decreasing service downtime by minimizing the number of IP reconfigurations that are required in case of WAN migration between the data centers. Moreover, we propose new data center selection algorithms that also aim to be proactive solutions to minimize IP reconfiguration time, resulting in minimizing the service

downtime. Two new metrics are proposed to indicate number of users that need IP reconfiguration and the total distance of IP reconfiguration time. We extended CloudSim to simulate and evaluate our proposed work for VM migration across the data centers on random workload. The experimental results show that our proposed algorithms have a significant reduction in terms of number of IP reconfigurations, and total distance than the other competitive VM selection algorithms.

5.2 Cost of Live VM Migration

The following subsections explain the LAN migration and WAN migration process. Moreover, metrics that are generally considered to measure the performance of live migration are defined.

5.2.1 LAN VM Migration

The main idea in LAN migration is to transfer memory state of VM as shown in Figure 5-1. Since we only need to migrate the VM memory state in the same LAN, we do not need any kind of network reconfiguration process where the VM IP address should not be changed. So, the destination host just only forces an ARP update within the broadcast domain. Thus, from now on, all traffic addressed to the VM is sent to the destination host, which itself forwards the traffic to the VM.

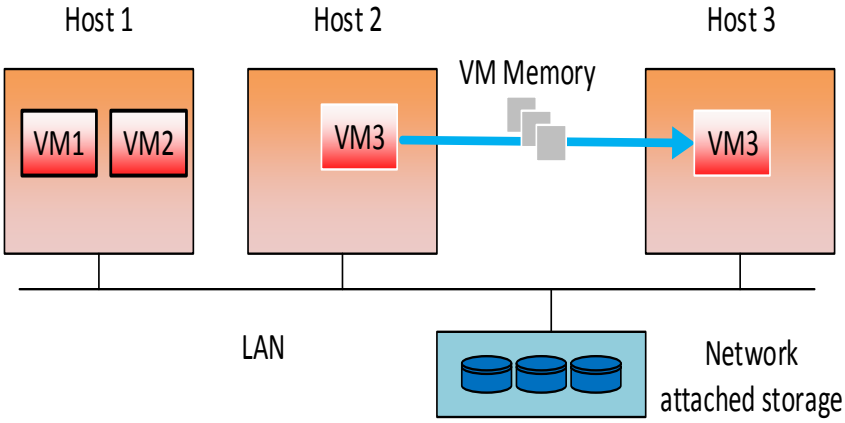

Figure 5-1: LAN Migration

Figure 5-2 explains the migration process in more detail [31]. The figure shows total migration time between t_0 and t_3, which represents the total time required to allocate a given VM on the destination host and deallocate this VM from the migrated host as notification of moving this VM. The figure shows the downtime between t_1 and t_2, which represents the portion of total migration time when the VM is not running, that is the time between pausing the VM on the source and resuming it on the destination.

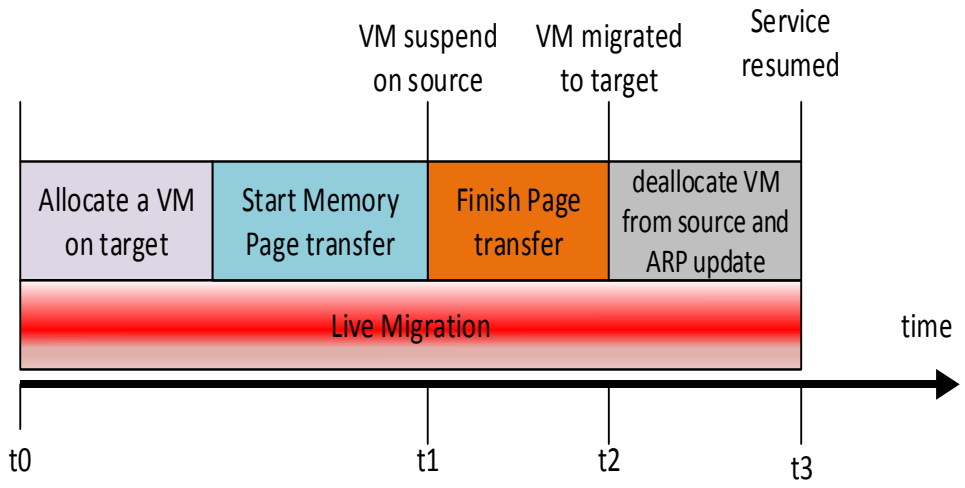

Figure 5-2: LAN Migration Process

5.2.2 WAN VM Migration

WAN migration is useful in many cases even though it has overhead related to network reconfiguration process and costs associated with storage migration. Users' geographical region might be one of the reasons for the VM WAN migration in order to keep VM closer to them, where large data centers or enterprises having computing infrastructure around the world to migrate VMs. Moreover, VM WAN migration could be applied in other cases to ensure load balancing, power saving [38], maintenance operation and upgrade a data center [30].

WAN VM migration transfers memory state and the state of local disks as well. Wide area migration uses the same concepts as legacy local migrations. However, one important factor disturbs an efficient deployment of wide-area migration across Internet Clouds. When a VM moves to a new subnet, a mobility solution or scheme should be applied to ensure that new connections are made seamlessly to its new IP address after the migration. Therefore, wide-area migration results into a mobility problem that may render the service unreachable unless network recovery is performed [29-31, 33, 36-39]. Also, it will increase the performance degradation. In fact, all the network recovery solutions still cause interruption of the service. Figure 5- 3shows VM migrations over WAN.

Figure 5-4 explains the migration process in more detail. The figure shows total migration time between t_0 and t_3, which now includes the network reconfiguration. It should be noted the downtime between $t_1$and t_2 for WAN VM migration is larger than that of LAN VM migration due to its file storage transfer. In practice, the service availability of VM does not depend only on the state of the VM (i.e., up or down) but

also on network connectivity [31]. As shown in Figure 5-4, there exists an additional time in WAN VM migration required for VM to resume its services.

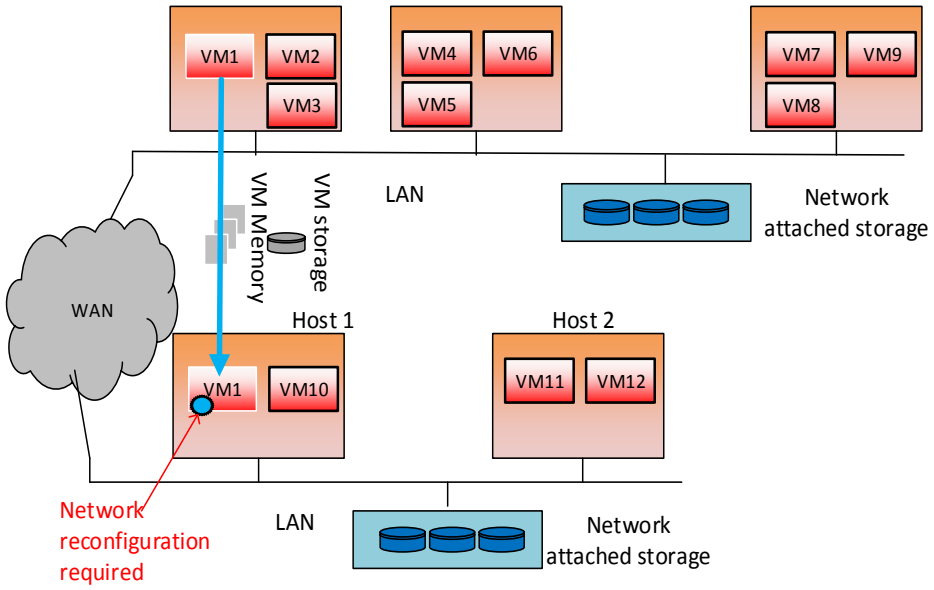

Figure 5-3: WAN Migration Process

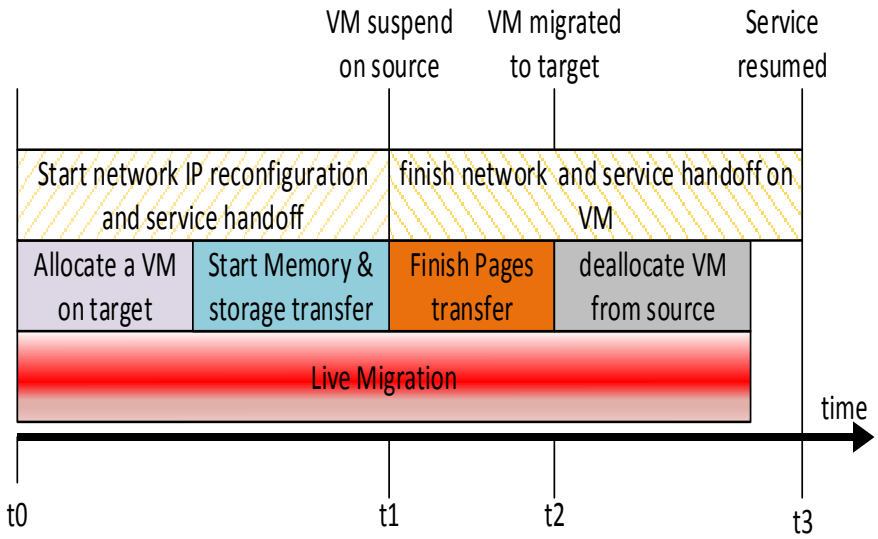

Figure 5-4: WAN Migration Process

The figure shows the IP reconfiguration time between t_2 and t_3, which represents the total time required to: 1) send a notification from the destination data center to the source data center indicating end the migration (that means the connected user on the

VMs have to connect to a new IP address), 2) send notification to the connected users indicating them to start connection with the new data center, and 3) the connection time between the users and the destination data center. The downtime and IP reconfiguration time that are required for VM to resume its services in WAN VM migration is called service downtime (t_3 - t_1) as indicated in Equation (5.1).

$$Service\ Downtime\ =\ Downtime + IP\ reconfiguration\ time \quad (5.1)$$

5.3 The Proposed System Model

In section 5.3.1, our proposed system architecture is explained. The VM selection algorithm and data center selection algorithms are explained in section 5.3.2.

5.3.1 System Architecture

The target system is an IaaS environment, represented by large-scale data centers. Each data center consists of a maximum of *J* heterogeneous hosts where each host contains multiple public VMs. Each VM can be connected to a number of users. Besides, each host and VM are characterized by the CPU performance metrics defined in term of MIPS, the amount of RAM and network bandwidth. The target system model is depicted in Figure 5-5, which is a modified version of the model described in [68].

Figure 5-5 shows the components of the proposed system model that provides proactive selection techniques to address IP reconfiguration issue in WAN live VM migration.

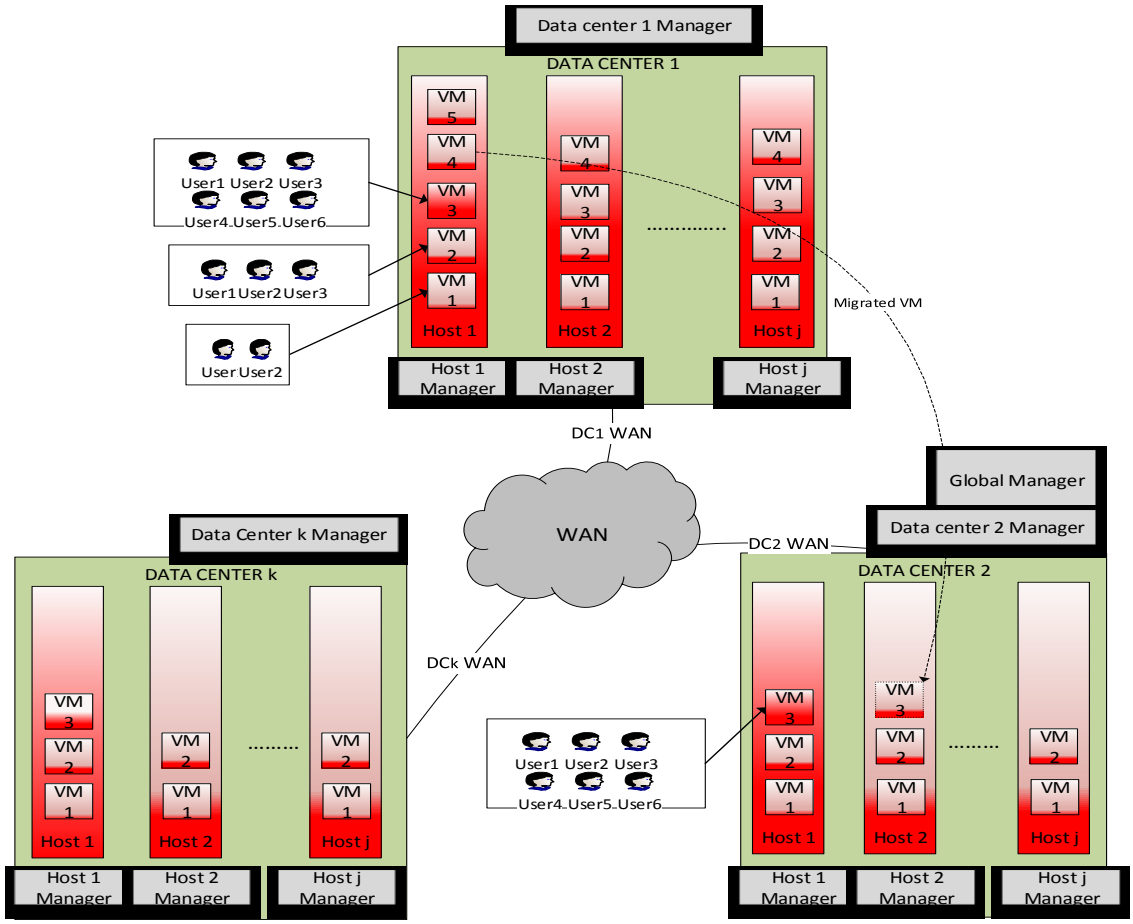

Figure 5-5: System Model

- Host Manager: a component that resides on every host for keeping continuous observation on CPU utilization of the node. It makes local decisions, such as deciding that the host is underloaded, or the host is overloaded and selecting VMs to migrate to other hosts.
- Data Center Manager: a component that resides on every data center and gathers information from host managers about the CPU utilization of its host, to manage the allocation of VMs locally or widespread and initiating LAN VM live migrations.
- Global Manager: a component that resides on one of the data centers and gathers

information from data centers managers and makes global management decisions, such as mapping VM instances to a data center manager and initiating WAN VM live migrations.

Host manager will check the CPU utilization status consistently for each host in its data center. If each host CPU utilization is less than a previously defined resource utilization threshold, then the system will be stable, and there is no need for LAN migration. When any host CPU overutilization is detected a VM LAN migration is triggered by data center manager between different hosts in the same data center to maintain the fairness and load balancing between the hosts. In contrast, when the data center CPU utilization is larger than a predefined CPU utilization threshold a VM WAN migration is triggered. Then the CPU utilization status of the data center should be sent to the global manager to select where to migrate the overloaded VMs based on the data received periodically from each data center to achieve the load balancing between the data centers as well.

5.3.2 The Proposed Work

Based on the proposed system model, the selection process algorithms can be divided into five parts: (1) Host underload/overload detection, (2) LAN / WAN migration, (3) VM selection, (4) Data center selection, (5) VM placement.

5.3.2.1 Host underload/overload detection

If a host is underutilized, then all the VMs from this host can be migrated in the same data center and the host will go to sleep/shutdown mode, or the host will be considered as a good candidate to receive the migrated VMs from the overloaded hosts in the future. On

the other hand, when a given host is overloaded some of its VMs must be selected to migrate from this host to other hosts in the same data center or even a different data center. In our experiments, the VM host detection algorithm used is (MAD) [42].

5.3.2.2 LAN/WAN migration

One of the reasons for WAN migration is when a data center is considered to be overloaded and one or more VM migration is required from data center under consideration. In our work, we assume the selected VMs always migrate to another data center to make extensive evaluation and performance analysis of the proposed VM selection and data center selection algorithms. In addition, we compare our proposed VM selection policy with the state of the art VM selection methods algorithms in case of WAN migration.

5.3.2.3 VM Selection

Once a host overload has been detected, it is necessary to determine which VMs are the best to be migrated from the host. We propose a new algorithm called Minimum Migration Time Maximum User Ratio (MMTMUR). The proposed algorithm takes the number of users in the selected VM to be migrated into its consideration, in order to obtain the minimum number of users that need IP reconfiguration due to WAN migration. A new parameter, User Ratio, is introduced which calculates the ratio between the number of users using the VM and the CPU utilization as indicated in Equation (5.2).

$$UserRatio_{vj} = \frac{CPU_{vj}}{Number\ of\ users\ of\ VM\ v\ of\ host\ j} \tag{5.2}$$

where CPU_{vj} is defined as given in Equation (5.3).

$$CPU_{vj} = \frac{TotalRequestedMIPS}{TotalMIPS\ for\ \text{VM v of host j}} \tag{5.3}$$

where CPU_{vj} is the amount of CPU currently utilized by the VM v of host j.

We consider both the VM migration time and User Ratio for migration between different data centers. The migration time can be estimated as the amount of memory used by the VM divided by the network bandwidth availability of that particular host [42]. Migration time is formalized in Equation (5.4).

$$TMig_{vj} = \frac{M_v}{B_j} \tag{5.4}$$

where $TMig_{vj}$ is the migration time of the VM v of host j, M_v is the amount of memory used by VM v, and B_j is the available bandwidth of the host j.

The pseudo-code of the VM selection for the overutilization case is presented in Algorithm 5-1. This algorithm first selects VMs with minimum migration time and sorts them in increasing order. Then, out of the first i selected VMs (in our case $i = 4$), the

Algorithm 5-1: Minimum Migration Time Maximum User Ratio (MMTMUR)

```
1   Input: OverloadedHost, VMs_RAM_values.
2   Output: a VM to migrate.
3   Max_UserRatio ← Min
4   selected_VM ← None
5   VMlist ← OverloadedHost.getVMlist()
6   VMlist.sortincreasing_VMs_RAM_values()
7   For (int i = 0; i < 4; i++)
8       VMlist2 [i] ← VMlist[i]
9   foreach VM in VMlist2 do
10      User Ratio = VM.getUserRatio
11      if User Ratio > Max_UserRatio then
12          Max_UserRatio ← User Ratio
13          selected_VM ← VM
14  return selected_VM
```

algorithm selects the VM with maximum user ratio, resulting in minimizing the number of users that need IP reconfiguration.

In our experiments, the state of the art VM selection algorithms used to compare with MMTMUR are presented in [40, 41, 52] including MC, MMT, MU and RS:

5.3.2.4 Data center Selection

Once the decision to migrate the VM from a given data center is made, it is necessary to find the most suitable data center to receive the migrated VM. We propose four data center selection algorithms:

- Data Center Random Selection (RSDC): selects a data center to receive the migrated VM randomly.
- Data Center Minimum Utilization (MUDC): selects a data center to receive the migrated VM based on the minimum utilization among data centers.
- Minimum IP Reconfiguration Time (MIPRT): selects a data center to receive the migrated VM based on the minimum total time needed to connect the migrated VM with its connected users.
- Minimum Utilization minimum IP Reconfiguration Time (MUIPRT): The algorithm first selects VMs with the minimum utilization and sorts them in increasing order. Then, out of the selected subset of VMs, the algorithm selects the VM with the minimum distance to minimize the live migration time.

5.3.2.5 VM Placement

VM placement refers to finding the most suitable hosts in the same data center or other data centers in order to receive the migrated VM. The VM placement algorithm method used is PABFD [42, 52].

5.4 Experimental Setup

In this section, we describe the simulation setup of our proposed approach. Then, the evaluation metrics will be described.

5.4.1 Simulation setup

We have simulated five data centers DC1, DC2, DC3, DC4, DC5 distributed in different geographical areas of (500,1000,1500,2000,2500) km respectively. Each data center contains 50 heterogeneous physical nodes of two types, half of the physical nodes are HP ProLiant ML110 G4 server (Xeon3040) and the other half consists of HP ProLiant ML110 G5 server (Xeon 3075). Each node is modeled to have two CPU cores with performance equivalent to 1860 MIPS for each core of the HP ProLiant ML110 G4 server, and 2660 MIPS for each core of the HP ProLiant ML110 G5 server. In addition, each node is modeled to have 1GB/s network bandwidth, 4GB of RAM and 50 GB of storage. Table 5-1 shows data centers configurations, Table 5-2 illustrates host types and Table 5-3 illustrates VM types.

The users submit requests for provisioning of 250 heterogeneous VMs, which are randomly distributed over four types similar to Amazon EC2 instance types: High-CPU Medium Instance (2500 MIPS, 0.85 GB), Extra Large Instance (2000 MIPS, 3.75 GB), Small Instance (1000 MIPS, 1.7 GB), and Micro Instance (500 MIPS, 0.633 GB). In

addition, each VM requires one CPU core with 2500, 2000, 1000 or 500 MIPS, 100 Mbit/s network bandwidth and 2.5 GB of storage.

Each VM runs an application with the variable workload, which is modeled to generate the utilization of CPU according to a uniformly distributed random variable. Each application has a length that determines the number of instructions. The application runs for 150,000 MI that is equal to 10 minutes of the execution on 250 MIPS CPU with 100% utilization. The interval of utilization measurements is every 5 minutes for 24 hours. Each VM is randomly connected to a maximum of 10 users with different randomly generated distance from 100 km to 1500 km.

Table 5-1: Data Centers Configurations

Data center	***Number of VMs***	***geographical areas (KM)***
D1	50	500
D2	50	1000
D3	50	1500
D4	50	2000
D5	50	2500

Table 5-2 Hosts Types

Host (Server) Type	***CPU Model***	***Cores***	***Frequency (MHz)***	***RAM (GB)***
HP ProLiant G4	Xeon3040	2	1860	4
HP ProLiant G5	Xeon 3075	2	2660	4

Table 5-3: VM types

VM Type (Instance)	***CPU (MIPS)***	***RAM (GB)***	***Maximum Connected Users***
High-CPU Medium	2500	0.85	10
Extra Large	2000	3.75	10
Small Instance	1000	1.7	10
Micro Instance	500	0.633	10

5.4.2 Performance Metrics

We considered six metrics to evaluate our system model. Four of them are previously defined in the literature, which are SLA violation, total energy consumption, total number of VM migrations that occur either for hotspot mitigation or for VM consolidation, and average SLA violation which describes how many times allocated resources are less than required resources. In this chapter, we propose two new metrics, which are the Number of IP reconfiguration and total distance for IP reconfiguration time. All of the six metrics are precisely defined below:

- Number of IP reconfiguration: higher number of users that need IP reconfiguration increases the network overload, and results in increased service downtime. Following equation can be used to calculate the number of IP reconfiguration during a given time interval for each data center.

$$Reconfigrations\ (P, t1, t2) = \sum_{j=1}^{J} \int_{t1}^{t2} \sum_{c=1}^{C} WMig_j(P, t) \qquad (5.5)$$

where P represents the current placements of VMs, J is the number of hosts, C is the number of connected users to the WAN migrated VM, $WMig_j(P, t)$ shows the number of WAN migrations of host j between time intervals t_1 and t_2 for the placement P.

- Total Distance: higher number of total distance increases the network overload, and results in increased service downtime. Following equation can be used to calculate the total distance during a given time interval for each data center.

$$Total\ Distance\ (P, t1, t2) = \sum_{j=1}^{J} \int_{t1}^{t2} WIPCD_j(P, t) \qquad (5.6)$$

where P represents the current placements of VMs, $WIPCD_j(P,t)$ shows the distance needed to measure the IP reconfiguration time for each WAN migrated VM of host j between time intervals $t_1 and\ t_2$ for the placement P.

$$WIPCD_j(P,t) = |\ x(dd,ds)|\ + \sum_{c=1}^{C} |\ x\ (ds,c)| + \sum_{c=1}^{C} |\ x(c,dd)| \tag{5.7}$$

where $x(dd,ds)$ represents the distance between the received data center and the migrated data center, C is the number of connected users to the WAN migrated VM. $x(ds,c)$ represents the distance between the migrated data center and the connected user. And $x(c,dd)$ represents the distance between connected user and the new location of the migrated VM.

5.5 Experimental Results

In this section, we first present the impact of our proposed algorithms on each data center separately and discuss our experimental results in comparison to the benchmark algorithms. We then show the impact of the proposed algorithms on the whole system.

5.5.1 Comparison with other benchmarks for each data center

We are interested in showing the impact of our proposed algorithms on each datacenter. We selected one performance metric, which is the number of IP reconfigurations, to compare our proposed VM selection algorithm MMTMUR with the existing VM

selection algorithms presented in [40, 41, 52] including MMT, MU, MC, and RS, among two proposed datacenter selection algorithms including, MIPRT and MUDC.

Figure 5-6 shows that data center DC5 has the least number of users that need IP reconfiguration for WAN migration in all the VM selection algorithms, whereas data center DC3 has the maximum number of users that need IP reconfiguration. It is completely obvious that the proposed VM selection algorithm MMTMUR significantly outperforms the other algorithms in terms of number of IP reconfiguration when the data center selection algorithm is MIPRT.

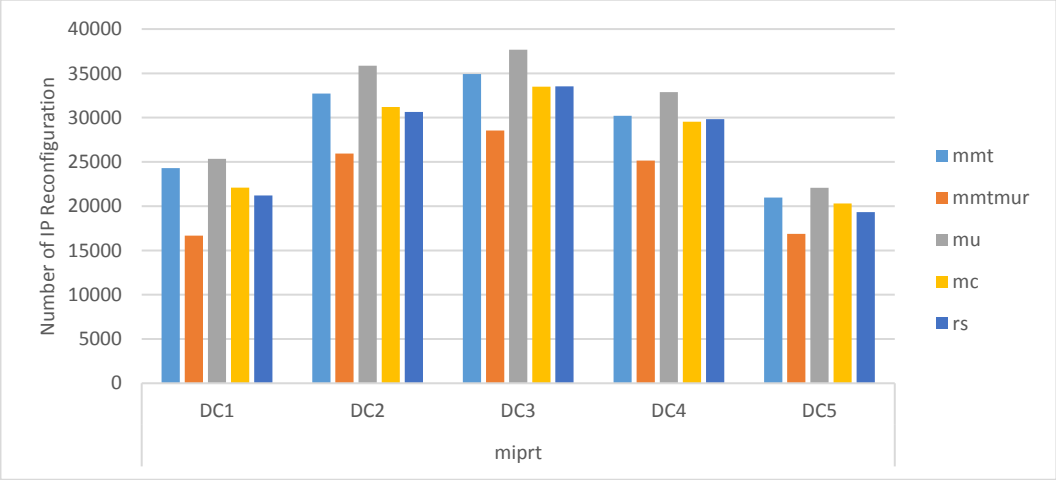

Figure 5-6: Number of IP Reconfiguration on each Data Center Using MIPRT Algorithm

Figure 5-7 shows another example of the selection process effect on each datacenter when the datacenter selection algorithm is MUDC. It is completely obvious that the proposed VM selection algorithm MMTMUR significantly outperforms the other algorithms in terms of number of IP reconfiguration when the datacenter selection is MUDC. The figure shows that all the datacenters have almost the same number of users that need IP reconfiguration for WAN migration for most of the VM selection algorithms, and is the least when the VM selection is MMTMUR.

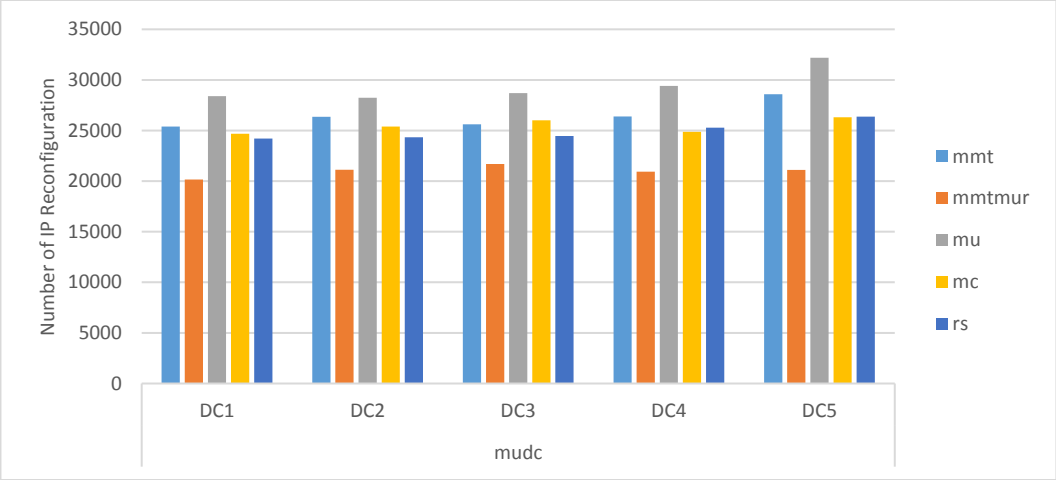

Figure 5-7: Number of IP Reconfiguration on each Data Center Using MUDC Algorithm

5.5.2 Comparison with other benchmarks in the whole system

We are further interested in comparing our proposed algorithms with the state-of-the-art algorithms. To perform this comparison, we employ the aforementioned six metrics in order to assess our results. Our comparison process is to study the algorithms' performance in the entire selection process which includes host detection, WAN/LAN migration, VM selection, data center selection and VM placement.

We compare the proposed algorithm, MMTMUR, with the state-of-the-art four VM selection algorithms, namely MMT, MU, MC, and RS. Besides, we investigate the impact of four proposed datacenter selection polices, namely MIPRT, MUDC, MUIPRT, and RSDC, on the VM selection algorithms.

From the simulation results depicted in Figure 5-8 and Figure 5-9 it is completely obvious that the proposed VM selection algorithm significantly outperforms the other algorithms in terms of number of IP reconfiguration and total distance in all datacenter selection algorithms.

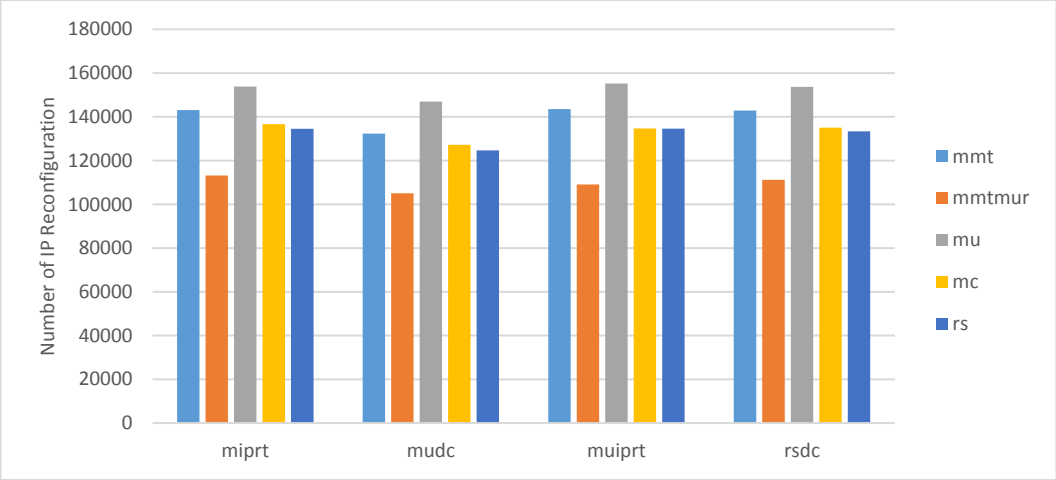

Figure 5-8: Number of IP Reconfiguration

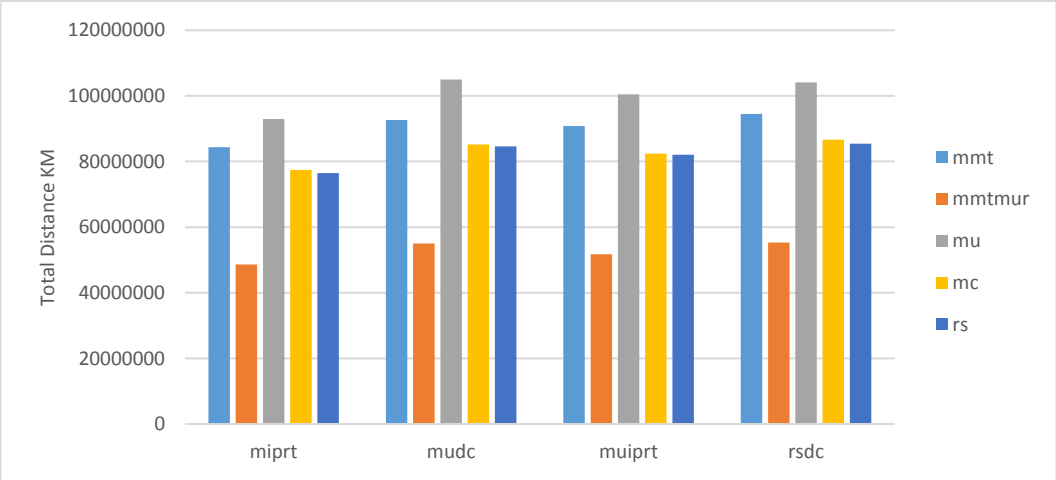

Figure 5-9: Total Distance

Figure 5-8 shows that our MMTMUR VM selection algorithm reduces number of IP reconfiguration metric by 20.93%, 26.44%, 17.18%, and 15.89% as compared to VM selection policies MMT, MU, MC and RS respectively when the data center selection is MIPRT, and by 24.01%, 29.74%, 19.02%, and 18.97% as compared to VM selection policies MMT, MU, MC and RS respectively when the data center selection is MUIPRT. Figure 5-8 also shows that MUDC outperforms the other data center selection algorithms in terms of number of IP reconfiguration.

Figure 5-9 shows that MMTMUR VM selection algorithm reduces total distance metric by 40.62 %, 47.61%, 35.43%, and 34.98% as compared to VM selection policies MMT, MU, MC and RS respectively when the data center selection is MUDC, and up to 43.04%, 48.51%, 37.24%, and 36.98% as compared to VM selection policies MMT, MU, MC and RS respectively when the data center selection is MUIPRT. From the simulation results depicted in Figure 5-9, it is completely obvious that MIPRT data center selection algorithm significantly outperforms the other algorithms.

Figure 5-10 shows that the proposed VM selection algorithm outperforms the other algorithms in terms of SLA violation in all the proposed data center selection algorithms except MUDC. It reduces SLA violation metric by 5.23%, 1.95%, 2.93%, and 2.31% as compared to VM selection policies MMT, MU, MC and RS respectively when the data center selection is MUIPRT. Moreover, from the simulation results depicted in Figure 5-10, it is obvious that MUDC data center selection algorithm has the best reduction of the SLA violation. Since MUDC selects the data center based on the minimum utilization, this lead to guarantee SLA for a long time more than the other data centers.

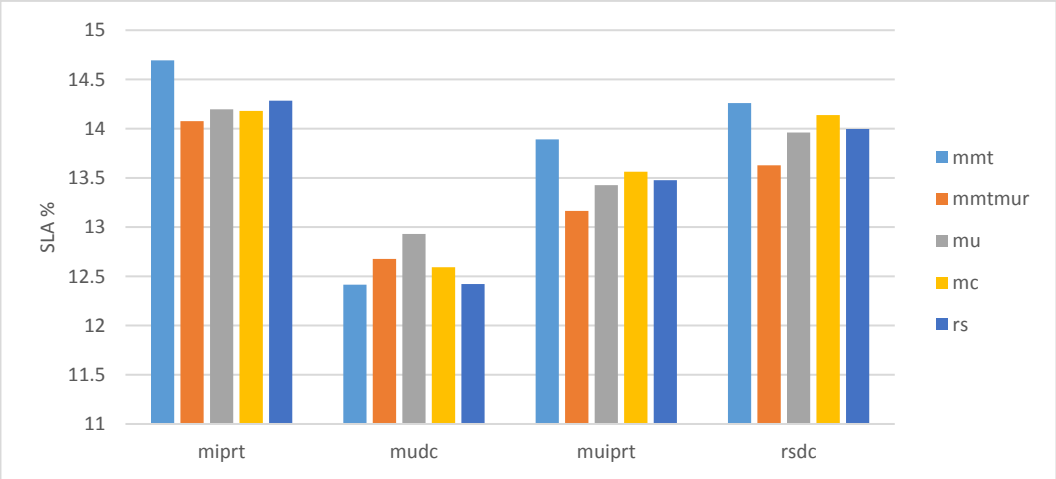

Figure 5-10: SLA Violation

From the simulation results depicted in Figure 5-11, it is completely obvious that the proposed VM selection algorithm outperforms the other algorithms in terms of number of VM migration metric in all the data center selection algorithms. Figure 5-11 shows that MMTMUR VM selection algorithm reduces number of VM migration metric up to 13.24%, 21.37%, 7.3%, and 7.46% as compared to VM selection policies MMT, MU, MC and RS respectively when the data center selection is MUIPRT. Figure 5-11 also shows that MUDC outperforms the other data center selection algorithms in terms of number of VM migration metric.

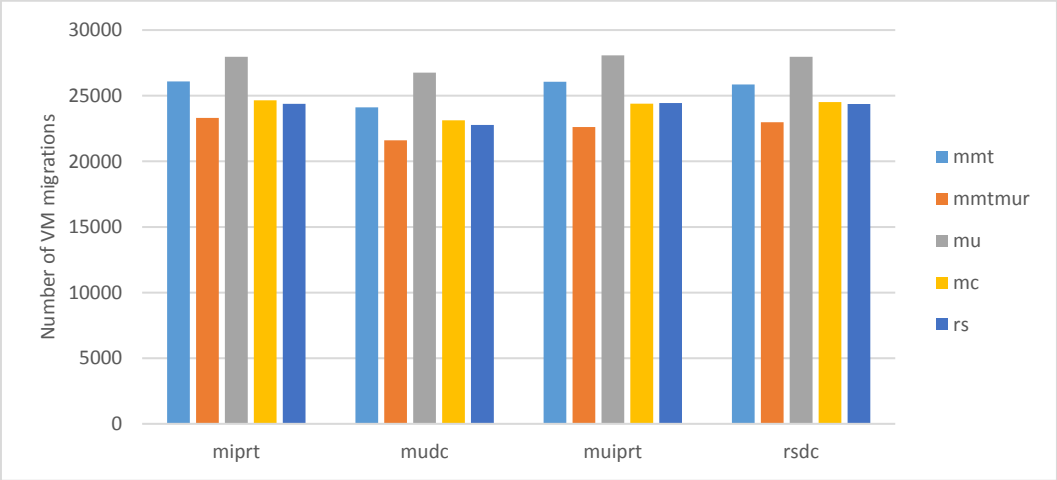

Figure 5-11: Number of VM Migration

Figure 5-12 shows that the proposed MMTMUR algorithm is slightly better than the other VM selection algorithm in terms of the energy consumption metric in all the proposed data center selection algorithms. It shows that MMTMUR VM selection algorithm reduces energy consumption metric by 10.98%, 16.94%, 8.91%, and 9.02% as compared to VM selection policies MMT, MU, MC and RS respectively, when the data center selection is MUIPTR. Moreover, from the simulation results depicted in Figure 5-

12, MIPRT data center selection algorithm outperforms the other algorithms in terms of energy consumption.

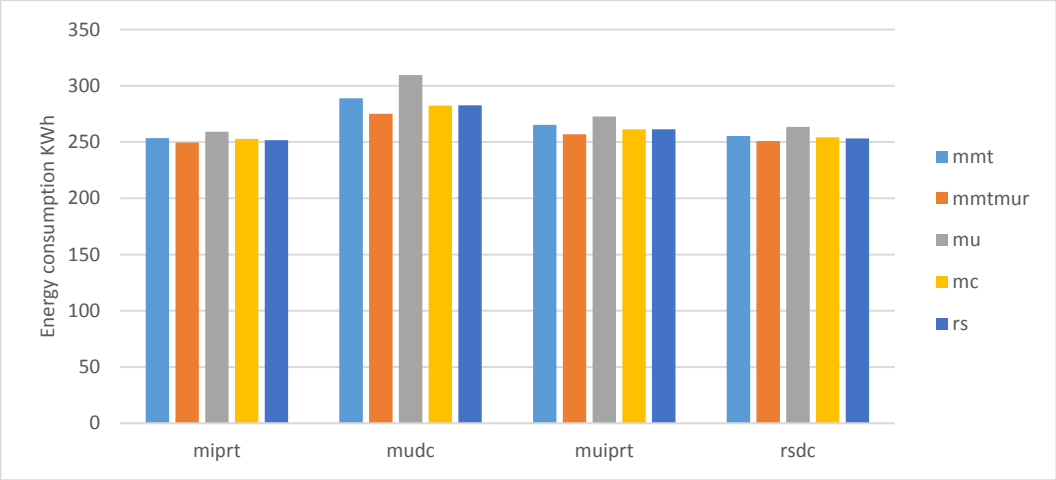

Figure 5-12: Energy Consumption

From the simulation results depicted in Figure 5-13, it is obvious that MMTMUR VM selection algorithm outperforms the other algorithms in terms of average SLA violation metric when the proposed data center selection algorithms are MUIPRT and RSDC. The figure shows that the best reduction of the average SLA violation is when the data center selection is MUIPRT and the VM selection algorithm is MMTMUR.

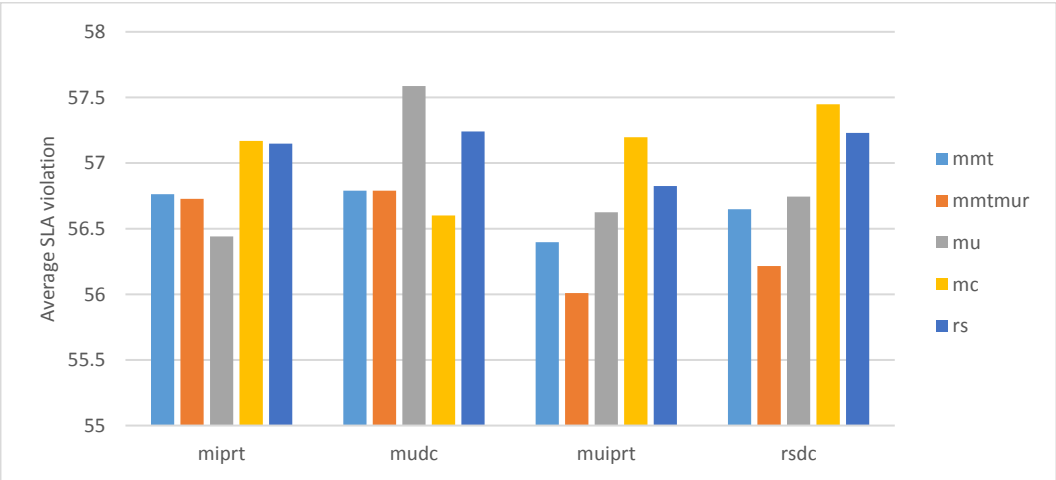

Figure 5-13: Average SLA Violation

5.6 Summary

We present a modified system model to indicate the number of users that need IP reconfiguration in case of WAN migration. This model has been proposed to consider neglected parameters and metrics that have an effect on live migration cost.

We present Minimum Migration Time Maximum User Ratio (MMTMUR) algorithm that aims to be a proactive solution for decreasing migration time by minimizing the time and the number of IP reconfigurations that are required in case of WAN migration between the data centers. The proposed algorithm takes the number of users in the selected VM to be migrated into its consideration, in order to obtain the minimum number of users that need IP reconfiguration due to WAN migration. The experimental results show that the proposed algorithm can significantly minimize the number of IP reconfigurations and IP reconfiguration time in terms of total distance as compared to the most commonly used MC, MMT, MU and RS algorithms, resulting in reduced service downtime and reduced network overhead.

We present Data Center Random Selection (RSDC), Data Center Minimum Utilization (MUDC), Minimum IP reconfiguration time (MIPRT), and Minimum Utilization Minimum IP reconfiguration time (MUIPRT) data center selection algorithms that aim to minimize the IP reconfiguration time, resulting in reducing service downtime.

Chapter 6

Conclusion and Future work

6.1 Concluding Remarks

Our research work is motivated by the necessity of improving service downtime, SLA violation and performance degradation in LAN/WAN migration. A lot of research has been done in the literature where many aspects need to be taken into our consideration in order to further amend the Quality of Experience (QoE) provided for the end users for higher user satisfaction in the system.

In order to optimize the resource utilization, we need to migrate VMs across hosts. Over time there are continuous changes in the status of the hosts. We find while migrating we cannot select the host considering the current state only, we need to consider various factors while selecting hosts from which we can shift VM or which we can consider as good candidate to receive VM.

In Chapter 3, firstly, we developed an algorithm called Median Absolute Deviation Markov Chain Host Detection Algorithm (MADMCHD). Unlike all available algorithms which depend on historical data to build probabilistic model that predict the future host load more efficiently. Its main goal was to improve SLA violation and to reduce VM

migrations. In our proposed algorithm three different states of given hosts are possible, namely (i) Underloaded (U), (ii) Normal Loaded (N) and (iii) Overloaded (O). First, we find state of all hosts that they are in which state using CPU current utilization value. Then our Markov detection algorithm starts working after collecting 10 observations based on probability observations. Hence based on our new algorithm instead of immediately migrating we can check whether migration is required or not. We consider a full system where Host Manager interacts with VMM Manager in order to initiate the VM migration process. CPU utilization upper and lower thresholds can be assigned either statistically using First Order-Markov Chain Host State Detection Algorithm (FOMCHSD) or dynamically using Median Absolute Deviation Markov Chain host Detection Algorithm(MADMCHD). We compared our algorithm with five host detection algorithms which are already implemented in Cloudsim for real workloads

Secondly, we developed Markov Power Aware Best Fit Decreasing (*MPABFD*) algorithm to enhance VMs placement process. The future candidate host load state is predicted to avoid overloaded state of that host after a short period. We combine the proposed algorithms in the selection process phases in the live migration for better performance, MadMCHD as a host detection algorithm, MPABFD as a VM placement algorithm, and some of the state of the art algorithms as a VM selection. We investigated the impact of these VM selection polices on the proposed model.

After host selection, in Chapter 4, we came up with two new VM selection algorithms namely Minimum VM Migrated Count and Minimum Migration time Minimum Migration Count to avoid frequent SLA violation on the same VM. MiMc (Minimum VM Migrated Count) –The Algorithm selects the VM to migrate from the

host overloaded based on the minimum number of VM migrated count. Minimum Migration Time Minimum VM Migrated Count (MmtMiMc)-The algorithm first selects VMs with minimum amount of RAM to minimize the live migration time and sorts them in increasing order. Then out of the selected subset of VMs the algorithm selects the VM with the minimum number of migration count. Along with that we are introducing two new metrics to compare with other existing VM selection algorithms. We have evaluated our proposed algorithms through CloudSim simulations on different planet lab real and random workloads and we are able to demonstrate that the proposed algorithms show significant reduction in maximum number of VM migrated count and degree of load balancing of VMs migrated count with other state of art algorithms.

Live VM migration across cloud data centers are useful for several cases despite the costs related to storage migrations and the overheads of network reconfiguration, such as maintenance and upgrades, and large data centers having computing infrastructure around the world that migrate VMs to follow the sun without affecting the end user experience. In Chapter 5, we modify the system model to provide proactive selection process techniques that reduce network reconfiguration problem in WAN live VM migration. This model has been proposed to consider neglected parameters and metrics that have an effect on live migration cost. We came up with a new VM selection algorithm, namely Minimum Migration Time Maximum User Ratio to be a proactive solution for decreasing service downtime by minimizing the number of IP reconfigurations that are required in case of WAN migration between the data centers. Moreover, we came up with new data center selection algorithms, namely Data Center Random Selection, Data Center Minimum Utilization, Minimum IP reconfiguration time,

and Minimum Utilization Minimum IP reconfiguration time that aim to minimize the IP reconfiguration time, resulting in reducing service downtime.

Two new metrics are proposed to indicate number of users that need IP reconfiguration and the total distance of IP reconfiguration time. We extended CloudSim to simulate and evaluate our proposed work for VM migration across the data centers on random workload. The experimental results show that our proposed algorithms have a significant reduction in terms of number of IP reconfigurations, and total distance than the other competitive VM selection algorithms.

6.2 Future Work

There is still more work to be done in cloud data center management. This list represents a few open topics.

- Run and apply the proposed algorithms with varying workloads which represents the cloud consumers' needs to realize the algorithms behaviors with it.
- Propose new measurement metrics that help the researcher to figure out the statistics data with different ways.
- Develop new datacenter selection algorithms based on different techniques that serves both cloud providers and cloud consumers which lead to reduce the operation cost in provider side which lead to reduce the cost services at cloud consumers side.
- Deploy and apply the proposed algorithms to run at real environment like Openstack which is an open source project and it's the best way to embraces the development of new features.

References

[1] P. Mell and T. Grance. “The NIST definition of cloud computing,” Version 15, 10-7 09. National Institute of Standards and Technology, Information Technology Laboratory, 2011.

[2] Google, “Google’s App Engine”, (2016), [online]. Available: https://cloud.google.com/ [Accessed: December 10, 2017].

[3] Amazon, “Amazon Elastic Compute Cloud (Amazon EC2)”, (2016), [online]. Available: aws.amazon.com/ec2/ [Accessed: December 10, 2017].

[4] Microsoft, “Microsoft Azure.”, (2016), [online]. Available: https://azure.microsoft.com/en-us/?b=16.11a [Accessed: December 10, 2017].

[5] IBM, “SmartCloud.” (2016), [online]. Available: http://www.ibm.com/cloud-computing/ [Accessed: December 10, 2017].

[6] HCL “Rise of The Cloud,” (2016), [online]. Available: http://www.hcltech.com/blogs/rise-cloud [Accessed: December 10, 2017].

[7] T. Veni and S. Bhanu. “A survey on dynamic energy management at virtualization level in cloud data centers,” Computer Science & Information Technology, pp. 107-117. 2013.

[8] VMWare, “vSphere ESX and ESXi,” (2016), [online]. Available: http://www.vmware.com/products/esxi-and-esx/ [Accessed: December 10, 2017].

[9] Microsoft, “Windows Virtual PC,” (2014), [online]. Available: https://support.microsoft.com/en-us/kb/958559 [Accessed: December 10, 2017].

[10] Xen, “Xen Hypervisor,” (2013), [online]. Available: http://www.xenproject.org/developers/teams/hypervisor.html [Accessed: December 10, 2017].

[11] Microsoft, “Hyper-V Server,” (2015), [online]. Available: https://technet.microsoft.com/library/hh831531.aspx [Accessed: December 10, 2017].

[12] Linux, “Kernel-based Virtual Machine (KVM),” (2016), [online]. Available: http://www.linux-kvm.org/page/Main_Page [Accessed: December 10, 2017].

[13] Oracle, "VirtualBox,” (2016), [online]. Available: https://www.virtualbox.org/ [Accessed: December 10, 2017].

[14] E. Bauer and R. Adams. “Reliability and availability of cloud computing,” John Wiley & Sons, 2012.

[15] K. Bilal, S. U. Khan and A. Y. Zomaya. “Green data center networks: Challenges and opportunities,” In Frontiers of Information Technology (FIT), 2013 11th International Conference, IEEE, pp. 229-234. 2013.

[16] M. Mishra, A. Das, P. Kulkarni and A. Sahoo. “Dynamic resource management using virtual machine migrations,” Communications Magazine, IEEE, vol. 50, no. 9, pp. 34-40, 2012.

[17] H. Jeong and B. Hong. “A management of resource ontology for cloud computing,” In Communication and Networking, pp. 65-72. Springer, 2011.

[18] B. Sotomayor, R. S. Montero, I. M. Llorente and I. Foster. “Virtual infrastructure management in private and hybrid clouds,” Internet Computing, IEEE, vol. 13, no. 5, pp. 14-22. 2009.

[19] P. Dreher, M. A. Vouk, E. Sills and S. Averitt. "Evidence for a cost effective cloud computing implementation based upon the NC state virtual computing laboratory model," Advances in parallel computing, high speed and large scale scientific computing, vol. 18, pp. 236-250. 2009.

[20] S. Averitt, M. Bugaev, A. Peeler, H. Shaffer, E. Sills, S. Stein, J. Thompson and M. Vouk. "Virtual Computing Laboratory (VCL)," In Proceedings of the International Conference on the Virtual Computing Initiative, pp. 1-6. 2007.

[21] M. A. Vouk, A. Rindos, S. F. Averitt, J. Bass, M. Bugaev, A. Kurth, A. Peeler, H. E. Schaffer, E. D. Sills and S. Stein. "Using VCL technology to implement distributed reconfigurable data centers and computational services for educational institutions," IBM Journal of Research and Development, vol. 53, no. 4, pp. 2-1. 2009.

[22] S. Mustafa, B. Nazir, A. Hayat and S. A. Madani, "Resource management in cloud computing: Taxonomy, prospects, and challenges," Computers & Electrical Engineering, vol. 47, pp. 186-203, 2015.

[23] P. T. Endo, G. E. Gonçalves, J. Kelner and D. Sadok. "A Survey on Open-source Cloud Computing Solutions," In Brazilian Symposium on Computer Networks and Distributed Systems, pp. 3-16. 2010.

[24] H. Elham, A. Lebbat and H. Medromi. "Enhance security of cloud computing through fork virtual machine," In Complex Systems (ICCS), 2012 International Conference, pp. 1-4. 2012.

[25] T. Rosado and J. Bernardino. “An overview of openstack architecture,” In Proceedings of the 18th International Database Engineering & Applications Symposium, pp. 366-367. 2014.

[26] V. F. Albor, J. Saborido, F. Gomez-Folgar, J. L. Cacheiro and R. G. Diaz. “DIRAC integration with CloudStack,” In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference, pp. 537-541. 2011.

[27] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield. "Live migration of virtual machines." In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 273-286. USENIX Association, 2005.

[28] M. R. Hines, U. Deshpande and K. Gopalan. “Post-copy live migration of virtual machines,” ACM SIGOPS operating systems review 43, no. 3, pp.14-26. 2009.

[29] R. Bradford, E. Kotsovinos, A. Feldmann and H. Schiöberg. “Live wide-area migration of virtual machines including local persistent state,” In Proceedings of the 3rd international conference on Virtual execution environments, pp. 169-179, 2007.

[30] T. Wood, K. Ramakrishnan, P. Shenoy and J. Van der Merwe. “CloudNet: Dynamic pooling of cloud resources by live WAN migration of virtual machines,” In ACM Sigplan Notices, vol. 46, no. 7, pp. 121-132, 2011.

[31] E. Silvera, G. Sharaby, D. Lorenz and I. Shapira. “IP mobility to support live migration of virtual machines across subnets,” In Proceedings of SYSTOR 2009, The Israeli Experimental Systems Conference, pp. 1-13, 2009.

[32] T. Hirofuchi, H. Ogawa, H. Nakada, S. Itoh and S. Sekiguchi. “A live storage migration mechanism over WAN for relocatable virtual machine services on

clouds," In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 460-465, 2009.

[33] H. Watanabe, T. Ohigashi, T. Kondo, K. Nishimura and R. Aibara. "A performance improvement method for the global live migration of virtual machine with IP mobility," In Proceedings of the Fifth International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2010), pp. 194-199, 2010.

[34] HP White Paper. "Live migration across data centers and disaster tolerant virtualization architecture with HP 3 PAR Cluster Extension and Microsoft Hyper-V," 2016.

[35] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni and A. Campi. "Clouds of virtual machines in edge networks," Communications Magazine, IEEE, vol. 51, no. 7, pp. 63–70, 2013.

[36] T. Wood, K. Ramakrishnan. "CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines," In IEEE/ACM Transactions on Networking, vol. 23, no. 5, pp. 1568-1583, 2015.

[37] U. Kalim, M. K. Gardner, E. J. Brown and W. Feng. "Seamless migration of virtual machines across networks," In Computer Communications and Networks (ICCCN), 2013 22nd International Conference, pp. 1-7, IEEE, 2013.

[38] S. Kuribayashi. "Improving quality of service and reducing power consumption with WAN accelerator in cloud computing environments," International Journal of Computer Networks & Communications (IJCNC), vol. 5, no.1, pp. 41-52, 2013.

[39] M. V. Bicakci and T. Kunz. “TCP-freeze: Beneficial for virtual machine live migration with IP address change?” In Wireless Communications and Mobile Computing Conference (IWCMC), 8th IEEE International, pp. 136-141, 2012.

[40] A. Beloglazov and R. Buyya, “Energy efficient allocation of virtual machines in cloud data centers,” 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 577-578, 2010.

[41] A. Beloglazov and R. Buyya, “OpenStack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds,” Concurrency and Computation: Practice and Experience, vol. 27, no. 5, pp. 1310-1333, 2015.

[42] A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers,” Concurrency and Computation: Practice and Experience, vol. 24, no. 13, pp.1397-1420, 2012.

[43] F. Farahnakian, P. Liljeberg and J. Plosila. “Lircup: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers,” In Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference, pp. 357-364. IEEE, 2013.

[44] K. Maurya and R. Sinha. “Energy conscious dynamic provisioning of virtual machines using adaptive migration thresholds in cloud data center,” Int.J.Comput.Sci.Mob.Comput, vol.3, no. 2, pp.74-82. 2013.

[45] S. S. Masoumzadeh and H. Hlavacs. “An intelligent and adaptive threshold-based schema for energy and performance efficient dynamic VM consolidation,” In Energy Efficiency in Large Scale Distributed Systems, pp. 85-97. Springer, 2013.

[46] L. Salimian, F. S. Esfahani and M. Nadimi-Shahraki. “An adaptive fuzzy threshold-based approach for energy and performance efficient consolidation of virtual machines,” Computing, pp. 1-20. Springer, 2015.

[47] A. Horri, M. S. Mozafari and G. Dastghaibyfard. “Novel resource allocation algorithms to performance and energy efficiency in cloud computing,” The Journal of Supercomputing, vol. 69, no. 3, pp. 1445–1461. Springer, 2014.

[48] E. Arianyan, H. Taheri and S. Sharifian. “Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers,” Computers & Electrical Engineering, vol. 47pp. 222-240. Elsevier, 2015.

[49] A. Beloglazov and R. Buyya, “Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints,” IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, pp.1366-1379, 2013.

[50] N. T. Hieu, M. Di Francesco and A. Yla-Jaaski. “Virtual machine consolidation with usage prediction for energy-efficient cloud data centers,” In Cloud Computing (CLOUD), 2015 IEEE 8th International Conference, pp. 750-757. 2015

[51] M. A. H. Monil and R. M. Rahman. “Implementation of modified overload detection technique with VM selection strategies based on heuristics and migration control,” In Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference, pp. 223-227. 2015.

[52] A. Beloglazov, “Energy-efficient management of virtual machines in data centers for cloud computing,” Ph.D. thesis, The University of Melbourne, 2013.

[53] S. Sohrabi and I. Moser. “The effects of hotspot detection and virtual machine migration policies on energy consumption and service levels in the cloud,” Procedia Computer Science, vol. 51, pp. 2794- 2798. Elsevier, 2015.

[54] K. Shahzad, A. I. Umer and B. Nazir. “Reduce VM migration in bandwidth oversubscribed cloud data centers,” In Networking, Sensing and Control (ICNSC), 2015 IEEE 12th International Conference, pp. 140-145. 2015.

[55] M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh and Q. Althebyan. “Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure,” Cluster Computing, vol. 18, no. 2, pp.919-932. Springer, 2015.

[56] X. Meng, V. Pappas and L. Zhang. “Improving the scalability of data center networks with traffic-aware virtual machine placement,” In INFOCOM, 2010 Proceedings IEEE, pp. 1-9. 2010.

[57] J. Xu and J. A. Fortes. “Multi-objective virtual machine placement in virtualized data center environments,” In Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom), pp.179-188. 2010.

[58] E. Feller, L. Rilling and C. Morin. “Energy-aware ant colony based workload placement in clouds,” In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing, pp. 26-33. 2011.

[59] F. Ma, F. Liu and Z. Liu. “Multi-objective optimization for initial virtual machine placement in cloud data center,” In Journal of Information & Computational Science, vol. 9, no. 16, pp. 5029–5038, 2012.

[60] D. Huang, D. Yang, H. Zhang and L. Wu. “Energy-aware virtual machine placement in data centers,” In Global Communications Conference (GLOBECOM), 2012 IEEE, pp. 3243-3249. 2012.

[61] G. Wu, M. Tang, Y. Tian and W. Li. “Energy-efficient virtual machine placement in data centers by genetic algorithm,” In Neural Information Processing, pp. 315-323. Springer, 2012.

[62] M. Tang and S. Pan. “A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers,” Neural Processing Letters, vol. 41, no. 2, pp. 211-221. Springer, 2015.

[63] C. T. Joseph, K. Chandrasekaran and R. Cyriac. “A Novel Family Genetic Approach for Virtual Machine Allocation,” Procedia Computer Science, vol. 46, pp. 558-565. Elsevier, 2015.

[64] S. K. Mandal and P. M. Khilar. “Efficient virtual machine placement for on-demand access to infrastructure resourcesin cloud computing,” International Journal of Computer Applications, vol. 68, no. 12, pp. 6–11, 2013.

[65] U. Mandal, P. Chowdhury, M. Tornatore, C. U. Martel, & B. Mukherjee “Bandwidth provisioning for virtual machine migration in cloud: Strategy and application,” IEEE Transactions on Cloud Computing, 2016.

[66] E. Fosler-Lussier, “Markov Models and Hidden Markov Models: A Brief Tutorial,” Technical Report (TR-98-041), International Computer Science Institute, Berkeley, California. 1998.

[67] A. Beloglazov, J. Abawajy and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” Future Generation Computer Systems, Elsevier, vol. 28, no. 5, pp. 755-768, 2012.

[68] A. Beloglazov and R. Buyya, “Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers,” Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, p. 4, 2010.

[69] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," IEEE International Conference on High Performance Computing & Simulation (HPCS'09), pp. 1-11, 2009.

[70] S. Ray and A. De Sarkar, “Execution analysis of load balancing algorithms in cloud computing environment,” International Journal on Cloud Computing: Services and Architecture (IJCCSA), vol. 2, no. 5, pp. 1-13, 2012.

[71] C. L. Dumitrescu and I. Foster, “GangSim: a simulator for grid scheduling studies,” IEEE international symposium on Cluster Computing and the Grid (CCGrid), vol. 2, pp. 1151–8, 2005.

[72] A. Legrand, L. Marchal and H. Casanova, “Scheduling distributed applications: the SimGrid simulation framework,” 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), pp. 138–45, 2003.

[73] R. Buyya and M. Murshed, "GridSim, a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," Concurrency and Computation: Practice and Experience; vol. 14, no. (13–15), pp. 1175–220, 2002.

[74] K.S. Park and V.S. Pai, "CoMon: a mostly-scalable monitoring system for PlanetLab," ACM SIGOPS Operating Systems Review, vol. 40, no. 1, pp. 65-47, 2006.

[75] N. Tziritas, C. Z. Xu, T. Loukopoulos, S. U. Khan and Z. Yu, "Application-aware workload consolidation to minimize both energy consumption and network load in cloud environments," 42nd IEEE International Conference on Parallel Processing (ICPP), pp. 449-457, 2013.

[76] X. Fan, W. D. Weber and L. A. Barroso, "Power provisioning for a warehouse-sized computer," ACM SIGARCH Computer Architecture News, vol. 35, no. 2, pp 13-23, 2007.