Adaptive resource optimization of three-tier web applications running on the cloud

Borhani, Amir Hossein

2018

Borhani, A. H. (2018). Adaptive resource optimization of three-tier web applications running on the cloud. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/83251

https://doi.org/10.32657/10220/46660

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

ADAPTIVE RESOURCE OPTIMIZATION OF THREE-TIER WEB APPLICATIONS RUNNING ON THE CLOUD

AMIR HOSSEIN BORHANI

School of Computer Scince and Engineering

2018

ADAPTIVE RESOURCE OPTIMIZATION OF THREE-TIER WEB APPLICATIONS RUNNING ON THE CLOUD

AMIR HOSSEIN BORHANI

School of Computer Scince and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

Acknowledgments

My foremost thank goes to my main supervisor, Associate Professor Terence Hung for all of his efforts, inspiration and support in my graduate study. I thank him for his understanding, optimistic guidance, and constructive comments on my work that helped to shape my research skills. As well, I wish to express my sincere gratitude to my first co-supervisor Associate Professor Bu-Sung Lee for his patience and professional feedback. Since the beginning of my PhD, he has motivated me to realize and develop my potential in research. I am also sincerely grateful to my second co-supervisor Dr. Zheng Qin for his scientific and impeccable guidance in my research period. Further, I must show my great gratitude to Dr. Xiaorong Li for her continued support, encouraging meetings and helpful suggestions. I will forever be thankful for her kind efforts for giving me the opportunity to pursue my PhD study in Singapore.

In addition, I would like to express my appreciation to Associate Professor Anis Yazidi from the University College of Oslo and Akershus of Applied Sciences, Oslo, Norway, for his valuable advice and encouragement. I gratefully acknowledge the scholarship support by the Agency for Science, Technology and Research (A*Star) of Singapore grant so that I can pursue my PhD in Singapore. I should be thankful to all my friends and colleagues in Nanyang Technological University and A*star Graduate academy for their numerous help. In particular, my great appreciation to Mrs Ng-Goh Siew Lai (Irene) for solving my technical computer problems quickly and helping me access the SCSE Custom Computing Resource to run some of my experiments. I would like to thank my family for understanding all the time I could not spend with them and their constant support and encouragement throughout my study. Last but not least, my deepest love and gratitude are devoted to my wife Marjan for being the foundation of my happiness in difficult times of my life and helping push me to the finish line. Thank you for working hard in Singapore to pay the tuition fee and living expenses while you suffered from severe eczema and headache.

Contents

Summary

Cloud Computing has been envisioned as a promising approach and dominant computing model in IT infrastructures. It employs the Virtual Machine (VM) technology to provide on-demand provisioning of resources on a pay-as-you-go base. This motivates enterprise application providers to adopt Cloud computing and outsource their infrastructures and computational needs. In particular, Cloud computing has become an attractive and promising platform for three-tier web applications. However, an inappropriate and inefficient resource management practice may negatively affect the Service Level Agreement (SLA) and the response time experienced by users, essentially under high load operating conditions. Furthermore, this may result in substantial amount of energy consumption in data centers, which consequently leads to a high operational cost.

This research consists of three major parts. The first part is a benchmark study that runs a three-tier web application on public cloud providers. It proposes *WPress* benchmark, which is based on the widespread blogging software, WordPress, as a three-tier web application, and implements an open source workload generator. Furthermore, a CPU micro-benchmark is utilized to investigate CPU performance of cloud-based VMs in greater detail. The main objective of this study is to evaluate and compare the average response time and operational cost of three-tier web applications running on commercial cloud providers. The small and large instance types of Amazon EC2, Microsoft Azure, and Rackspace Cloud are evaluated. Based on the experimental results it is found that Rackspace and Microsoft Azure are the preferred cloud solutions for small and large instance types, respectively. Furthermore, it is noticed that average response time has substantial fluctuations for large instances that can lead to significant SLA Violation (SLAV) for high load.

Cloud service providers are penalized if the service level agreement is violated. This may result in the loss of revenue. It is hence important for service providers to minimize SLAV as much as possible. To address this, the second part of the thesis introduces a network-aware VM migration algorithm to minimize the average SLAV of the system. The algorithm considers steady-state traffic condition to minimize the negative effect of migration on other flows. The Network Gain (NG) is calculated for candidate VMs and the VM with the maximum NG is selected. The simulation results in CloudSim show that the suggested algorithm yields significant performance improvements.

The increase in computing capacity and communication units in modern data centers results in the high energy consumption and operational cost. This negatively affects the environment and the cost of using cloud. Therefore, it is crucial to design new approaches for saving the energy consumption in the data centers. The third part of this thesis is to address this issue by extending the network-aware algorithm with energy-awareness capabilities to minimize the energy consumption while maintaining the SLA. In addition to NG, Power Gain (PG) is calculated for each candidate VM and two lists are created for each congested link: NG list and PG list. The VM with the lowest sum of the rank is selected. An extensive simulation study in CloudSim presents the effectiveness of the proposed approach.

List of Figures

List of Tables

List of Abbreviations

α-MCNU	α-Minimum Migration Cost Network Utilization
A*Star	Agency for Science, Technology and Research
ACES	Automated Controller for Energy-aware Servers
AHP	Analytic Hierarchy Process
App-tier	Application-tier
APP-VM	Application Server VM
AR	Aggregated Request
AWS	Amazon Web Services
BaaS	Benchmark-as-a-Service
BFH	Best Fit Host
BFV	Best Fit VMs
CARE	Cloud Architecture Runtime Evaluation
CDN	Content Delivery Network
CI	Confidence Interval
CRM	Customer Relationship Management
CSO	Cat Swarm Optimization
CVM	Container Virtual Machine
DaaS	Database-as-a-Service
DBMS	Database Management System
DB-tier	Database-tier
DB-VM	Database Server VM
DCN	Data Center Network
DENDIST-FM	Dynamic Reroute with Flow Migration
DNA	Deoxyribonucleic Acid
EC2	Elastic Compute Cloud
ECU	EC2 Compute Unit
ERP	Enterprise Resource Planning
ETA-VMM	Energy-and-Topology Aware VM Migration
FCFS	First-Come First-Serve
FFD	First Fit Decreasing
GA	Genetic Algorithm
GB	Gigabyte
GbE	Gigabit Ethernet
GL	Group Leader
GM	Group Manager
HPC	High Performance Computing
HSP	Host Selection Policy
HVM	Hardware Virtual Machine

IaaS	Infrastructure-as-a-Service
IIS	Internet Information Server
IQR	Interquartile Range
kWh	kilowatt-hour
LC	Local Controller
LIPHostSort	Least Increased Power with Host Sort
LR	Local Regression
MA	Multi-Agent
MAD	Median Absolute Deviation
MBFD	Modified Best Fit Decreasing
Mbps	Megabit per second
MC	Maximum Correlation
MCC	Minimum Correlation Coefficient
MCMC	Maximum Correlation with Migration Control
MCRVMP	Min Cut Ratio-aware VM Placement
MIPS	Million Instructions Per Second
MLB	Migration with Load-Balancing Threshold
MM	Migration Manager
MST	Migration with Static Threshold
MWC	Monotonic Write Consistency
NA	Network-Aware
NAS	Network Attached Storage
NC	Network Cloudlet
NG	Network Gain
NIST	National Institute of Standards and Technology
NM	No Migration
NP	Network Priority
NPB	NAS Parallel Benchmark
NS2	Network Simulator
NTU	Nanyang Technological University
NUS	National University of Singapore
OLTP	On-Line Transactional Processing
OS	Operating System
PaaS	Platform-as-a-Service
PB	Power Balancer
PG	Power Gain
PNA	Power and Network-Aware
PP	Power Priority
PS	Power Saver
PVA	Peer VM Aggregation
PVM	Paravirtualized Virtual Machine
R	Read workload
R/W	Read/Write workload
RFDCN	Removed From the DCN
RLR	Robust Local Regression
RNA	Ribonucleic Acid
RSST	Resultant Steady State Traffic
RT	Retransmitted Traffic

RYWC	Read Your Writes Consistency
S3	Simple Storage Service
SaaS	Software-as-a-Service
SCSE	School of Computer Science and Engineering at NTU
SDN	Software Defined Network
SEO	Search Engine Optimization
SGT	Singapore Time
SLA	Service Level Agreement
TAVMS	Topology-Aware Virtual Machine Selection
THR	Static CPU Utilization threshold
THR-RS	Threshold with Random Selection
ToR	Top-of-Rack
Ubench	Unix benchmark utility
UUT	Upper Utilization Threshold
VC	VM Clustering
vCPU	virtual CPU
VM	Virtual Machine
VMM	Virtual Machine Manager
VPC	Virtual Private Cloud
VSP	VM Selection Policy
W	Write workload
WAN	Wide Area Network
WEB-VM	Web Server VM
WIPS	Web Interactions Per Second
YCSB	Yahoo! Cloud Servicing Benchmark

Chapter 1

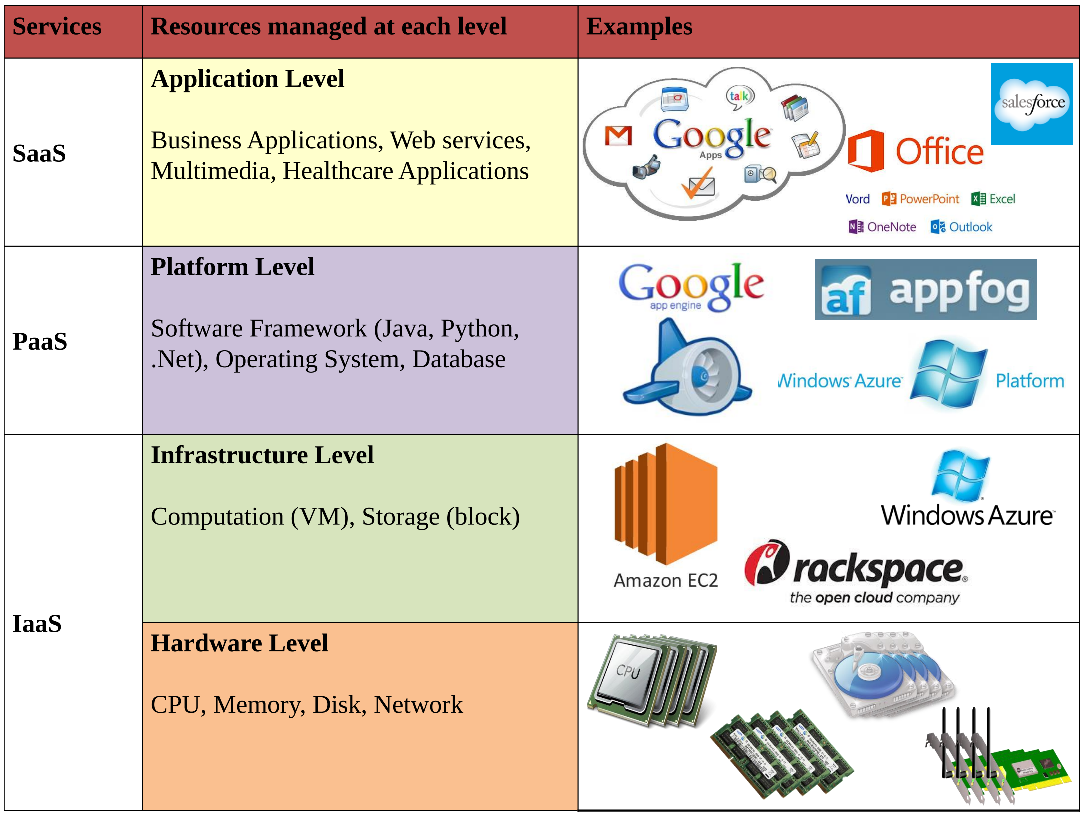
Introduction

This chapter is an overview of basic terms and terminologies that are employed in this thesis. Firstly, I describe the cloud computing architecture including different types of cloud and available cloud services. Next, I explain the virtualization technology and illustrate existing types of virtualization. After that, I explain the Live migration in data centers and show existing approaches.

1.1 Overview of cloud computing

Cloud computing is an emerging information technology which provides on-demand shared resources and services over the Internet on the pay-as-you-go basis with the predicted market size of $228 billion by 2026 [3]. The primary goal of cloud computing model is to establish a proficient utilization of distributed resources (i.e. computing, storage and network) with high availability, security and cost effectiveness. The lack of a standard definition of cloud computing has led to small amount of indecision and confusion. There have been several research works on standardizing the definition of cloud computing [4–16]. In this thesis, the definition of cloud computing proposed by The National Institute of Standards and Technology (NIST) [16] is adopted because, it is able to cover the major features of cloud computing: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Indeed, the key reason for existence of various impressions of cloud computing is that it serves as a basic model to bring together a set of existing technologies such as grid computing, utility computing and virtualization each of which has their own definition and characteristics. Essentially, cloud computing leverages the aforementioned technologies to run a business in new way which is more effective and efficient to meet the technological and economic requirements of today's demand for information technology [17].



Services	Resources managed at each level	Examples
SaaS	**Application Level** Business Applications, Web services, Multimedia, Healthcare Applications	Google Apps, salesforce, Office, Vord, PowerPoint, Excel, OneNote, Outlook
PaaS	**Platform Level** Software Framework (Java, Python, .Net), Operating System, Database	Google app engine, af appfog, Windows Azure Platform
IaaS	**Infrastructure Level** Computation (VM), Storage (block)	Amazon EC2, Windows Azure, rackspace the open cloud company
	Hardware Level CPU, Memory, Disk, Network	CPU

Figure 1.1: The architecture of cloud computing.

1.1.1 Cloud Computing Architecture

The architecture of cloud computing environment (refer to Figure 1.1) consists of 4 layers: 1) the hardware layer, 2) the infrastructure layer, 3) the platform layer, and 4) the application layer. While the modular architecture enables each layer to behave and evolve separately, it facilitates each layer to communicate with its successor or predecessor. At the highest level of the hierarchy, the application layer refers to the actual software and applications that are available over the cloud including business, web services, multimedia, health care, etc. The next layer, known as platform layer, provides the software frameworks (e.g. Java, Python, .Net, etc.), operating system (OS) and database.The purpose of platform layer is to eliminate the complexity of establishing and maintaining the infrastructure usually associated with developing and launching an application. The subsequent layer is called infrastructure, also known as virtualization layer. This is an essential part of cloud computing that uses the virtualization technologies (e.g. Xen [18], VMware [19], and KVM [20]) and provides dynamically scalable virtual resources in the form of Virtual Machine (VM), storage blocks and network bandwidth. The lowest layer is the hardware, also known as data center, layer that consists of all physical resources being used in the cloud environment including physical machines, network devices, power source and cooling systems [17].

A typical data center contains hundreds or thousands of servers. Figure 1.2 illustrates a typical building block of a data center. The physical servers (e.g. web, application and database servers) are organized in multiple racks which in turn are interconnected via the data center network (DCN). Upon receiving a new request, the content switches and the load balance devices choose the appropriate server to direct the request to, and control the path taken by packets in the network. Quite possibly, this server needs to communicate with many other servers to fulfill the request. A good example is posting a new photo into your personal web-log in which web, application and database servers are involved.

The DCNs are usually established in either two- or three-tiers of switches or routers. The DCN depicted in Figure 1.2 has a canonical fat-tree 3-tiered design including edge, aggregation and core tiers. In the edge tier, also known as access tier, each rack is connected a 1 Gbps Top-of-Rack (ToR) switch. In the aggregation tier, ToR switches are connected to 10 Gbps switches so-called aggregation switches. Finally, the core tier is to interconnect aggregation switches either via 10 Gbps or 100 Gbps links (a bundle of 10 Gbps links [21]). The fat-tree architecture is very promising in terms of scalability, fault-tolerance and simplicity of the interconnection.

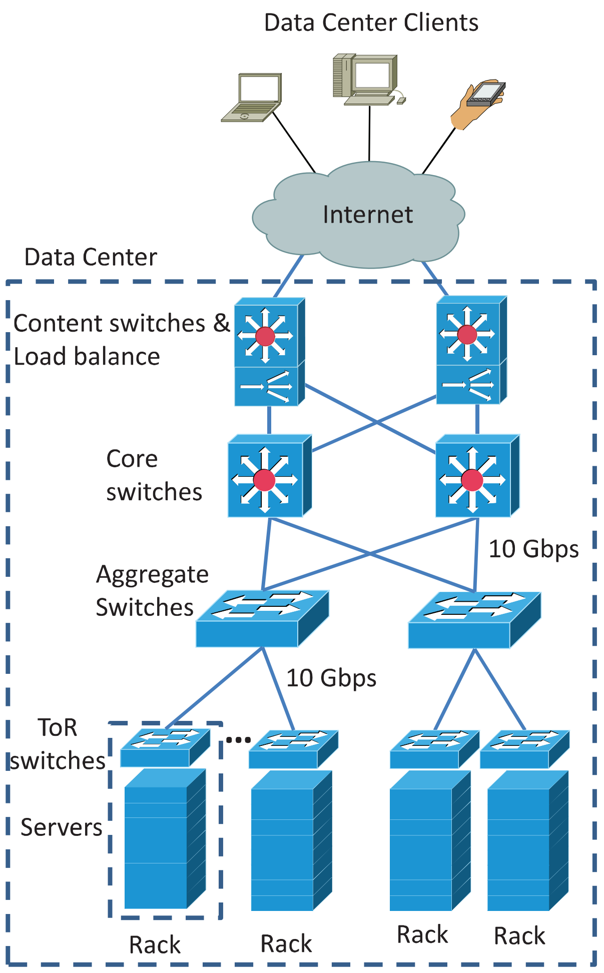


Figure 1.2: The architecture of data center. (Adopted from [22])

1.1.2 Cloud Computing Services

Cloud computing promises to provide its resources as a subscription-based services. In other word, the four levels of resources depicted in Figure 1.1 (i.e. hardware, infrastructure, platform, and application) are delivered on an on-demand basis with pay-as-you-go business model. In practice, cloud services are classified in three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [9–15, 17, 23–26] (readers are referred to figure 1.1).

(i) *Infrastructure as a Service (IaaS)*

IaaS, is one of the most flexible services offered by cloud providers. This is a systematic approach to deliver a variety of infrastructure resources (i.e. compute, storage and network) as well as several services (e.g. load balancing and content delivery networks (CDNs)) over the Internet. For instance, Amazon Elastic Compute Cloud (EC2) offers compute resources while Amazon S3 and EBS offer storage resources. IaaS server hosting service is the dominant solution for enterprises and organizations in the current market environment with which, they can run their businesses from scratch by renting a desired number of virtual machines called instances each of which with a predefined amount of physical resources being assigned from the hardware level shown in figure 1.1. In particular, IaaS has the following substantial benefits:

(a) No CAPEX (no upfront heavy investment)

(b) Minimal delay in getting started

(c) Save time and effort to procure, setup and maintain the resources as these services are taken care of by the providers

According to the information provided by Dan [27], the most popular IaaS providers are Amazon AWS, Windows Azure, Google Compute Engine, Rackspace Open Cloud, IBM SmartCloud Enterprise and HP Enterprise Converged Infrastructure.

(ii) *Platform as a Service (PaaS)*

PaaS, is specifically designed for software developers. It provides several programming platforms (e.g. Java, Ruby, PHP, Python, Go, Node, Scala, Erlang, .NET, etc.) and database applications (e.g. MySQL, PostgreSQL, Redis, Microsoft SQL Server, etc.). All developers need to do is just selecting their favorite platform and starting their application development, testing and deployment without having to worry about the hardware that is being used. The major advantage for PaaS customers is that they are not required to invest in expensive infrastructure to make use of the application provided as a service. This is because of the fact that PaaS providers charge customers only for the portion of time that they use the platform. Furthermore, PaaS is considered as an effective IT solution in terms of availability and fault tolerance because normally multiple copies of user's data are stored on provider's data-centers. Moreover, PaaS offers dynamic scalability where allocated resources scale up or down automatically based on application demand [12, 13, 15, 24]. Some of the popular PaaS providers are Google App Engine, Microsoft Azure platform, and AppFog. [28].

(iii) *Software as a service (SaaS)*

Perhaps, SaaS is one of the most popular cloud services on which enterprise applications, which are already installed and running on the cloud, are leased on-demand [29, 30]. The applications offered in SaaS layer may have been developed and deployed on the lower levels, such as PaaS and IaaS, of the cloud service hierarchy. In particular, SaaS changes traditional software usage to a subscription-based model to reduce the user cost of infrastructure [30]. SaaS has been adapted in fast pace for many business applications including customer relationship management (CRM), enterprise resource planning (ERP) [31]. Some of the well-known SaaS providers are: Google App (Google Doc, Google Mail, Google Spreadsheets), Microsoft Office 365 and SalesForce.com [32].

1.1.3 Cloud Types

Depending on infrastructure ownership and management, cloud computing are classified into three major types: Private Cloud, Public Cloud and Hybrid Cloud [17, 30]. According to the survey conducted by RightScale in 2016 [33], almost 95% of technical professionals across a broad cross-section of organizations adopted some types of cloud solutions (see Figure 1.3).

(i) *Public Cloud*

In public cloud, the infrastructure and virtualization software are owned by a third-party organizations commonly known as public cloud providers, such as Amazon EC2, Microsoft Azure, Rackspace Cloud and Google Apps. They offer cloud services over Internet to be accessible for general public. Public clouds,

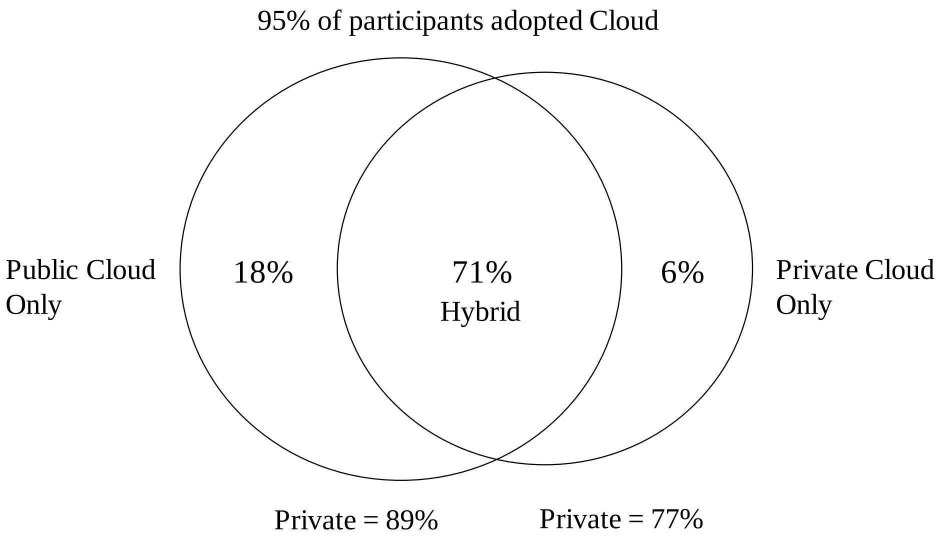


Figure 1.3: The 2016 survey results, conducted by RightScale, in terms of popularity of cloud types (adopted from [33])

also referred as hosted cloud [8], usually have several data centers distributed in different countries and regions to meet resource demand of thousands of million users each with different QoS requirement. The Service Level Agreement (SLA) is a contract negotiated and agreed between a customer and a service provider of the level of service to be provided [34]. Public cloud offers services in a pay-as-you-go business model. This is a very cost-effective computing solution particularly for small organizers or individuals because they do not need to purchase any infrastructure or software [35, 36]. Nonetheless, public clouds are unable to provide users a fine-grained control over allocated resources [12, 24]. In other words, customers are not allowed to alter the instance details such as number of virtual CPUs (vCPUs), memory, storage, etc. To address this issue, cloud providers tend to develop a reasonable variety of instance types to meet customer demands. For example, at the time of writing this report Amazon EC2 offers 55 types of instances which are classified into five categories: general purpose, compute optimized, memory optimized, accelerated computing, and storage optimized [37]. Table 1.1 illustrates the number of instances being offered coupled with a few sample use cases for each category.

(ii) *Private Cloud*

Table 1.1: The instance types offered by Amazon EC2, last viewed on January 2017

Category	No. of instance	Sample use case
General Purpose	17	web applications, code repositories
Compute Optimized	10	web servers, distributed analytics
Memory Optimized	13	big data processing engines, data mining
Accelerated Computing	7	machine learning, 3D application streaming
Storage Optimized	8	NoSQL databases, data warehousing

Private cloud is devoted for internal use of organizations. In fact, private cloud providers are mostly large companies or government departments, that are mainly interested to have more security and control over its data [30]. Private cloud can be set up within an internal data center of an organization or outsourced to a third party to operate. The Microsoft Azure Government [38] is a good example of the United States Federal government-specific private cloud that is outsourced to Microsoft. Essentially, private clouds are able to provide many of the features of public clouds such as elasticity and being service based [36]. Some of the most popular private cloud solutions are CloudStack [39], OpenStack [40] and Eucalyptus [41].

(iii) *Hybrid Cloud*

The hybrid cloud is formed by an integration of private and public clouds [42]. In this model, high flexibility of resource provisioning offered in private cloud is combined with cost effective infrastructure of public clouds. Essentially, services with high security demands are kept within the control of the organization and the remaining services can be safely outsourced to public cloud providers. Furthermore, if internal demands exceed the available resources in the private cloud, more resources can be leased from public cloud providers to meet an organization's overall business objectives. Some good examples of hybrid cloud are Virtual Private Cloud (VPC) services offered by Google [43] and Amazon [44].

1.2 Virtualization Technology

Virtualization technology has a long history in computer science, dating back to the 1960s with IBM's System/360 (S/360) and System/370 (S/370) [45, 46]. The term *Virtualization* is used to describe the abstraction of computing resources required to complete a service and the underlying physical infrastructure (CPU, storage, network, memory). Virtualization is the core technology used in cloud computing that enables multi-tenancy by providing shared, scalable resource platform for all tenants [47]. In particular, a special type of operating system, known as hypervisor or Virtual Machine Manager (VMM), is running on top of the physical hardware infrastructure. Hypervisor is capable of making resource allocation completely independent of operating system level. By doing this, the hypervisor is able to execute multiple operating systems and their associated applications in smaller computing units, known as Virtual Machine (VM), each of which runs on a single physical server. It is the job of hypervisor to hide all the complexity from end-user, so that from the users' point of view each VM is perceived as if it is the only machine on the physical infrastructure [48]. Figure 1.4 depicts an example of a virtualized cloud environment with two servers each of which runs two guest operating systems (OSs). Tenants have access to the applications through their web browsers without knowing anything about where they are located [49].

The state-of-the-art virtualization technologies are generally classified into three categories: *Paravirtualized Virtual Machine (PVM)*, *Hardware Virtual Machine (HVM)*, and *Container Virtual Machine (CVM)* [52]. The main similarity between PVM and HVM is the use of hypervisor between VMs and underlying physical resources (Figure 1.4). Hence, both technologies are able to ensure security and confidentiality of running VMs. However, they differ with regards to the modification of guest OS. In HVM, the guest OSs can be run without any modification on top of the hypervisor as if they are directly interacting with the bare metal hardware. But, in fact the guest OSs are running as user-level processes on top of the hypervisor. Therefore, the hypervisor needs to emulate certain functionalities of the guest OS that can only

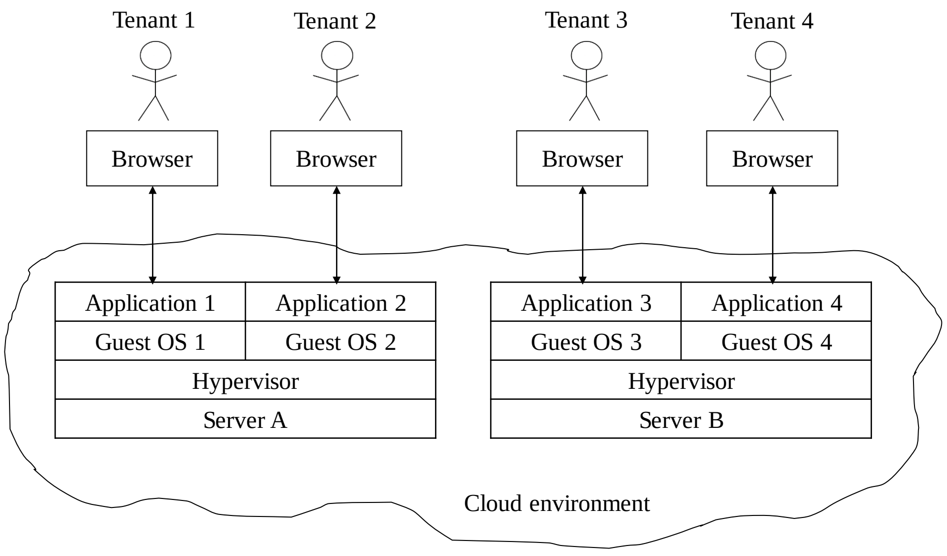


Figure 1.4: An example of hypervisor-based virtualization in cloud computing environment (adopted from [50, 51])

be run in the privileged mode. In the PVM design, instead, the binary code of the guest OS is modified such that the guest OS become aware of the virtualized environment and hypervisor. Hence, in PVM the hypervisor does not need to emulate certain functionalities of the guest OS [52–54]. VMware [55], is a well known product for HVM, and Xen [18] is a widely used project of PVM. To name a few commercial cloud providers, Amazon AWS [44] uses Xen hypervisor, and vCloud Air [56] employs VMware. CVM, also known as software virtualization, is a lightweight alternative to

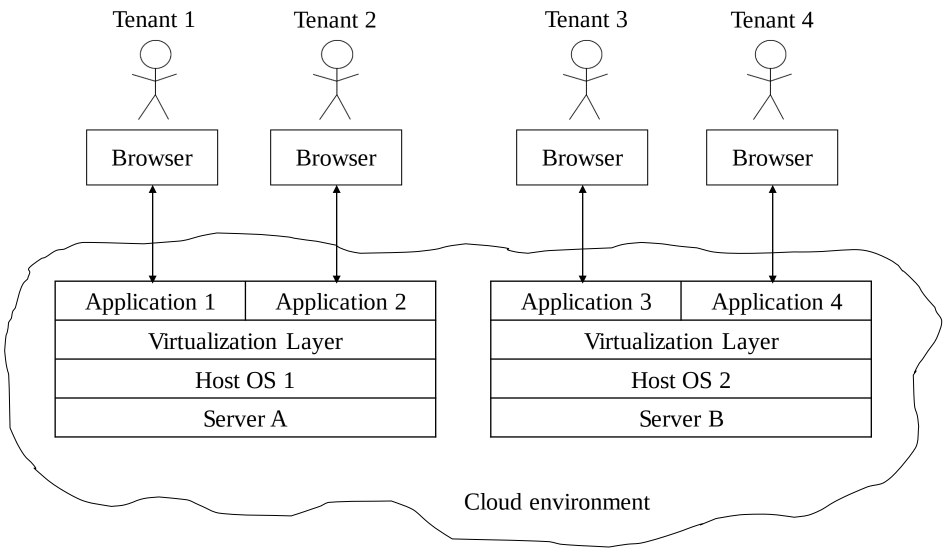


Figure 1.5: An example of container-based virtualization in cloud computing environment (adopted from [50, 51, 57])

the hypervisor-based virtualization technologies such as HVM and PVM. In particular, CVM is an OS-level virtualization technology that enables applications to run on distributed resources without launching a separate VM for each application [57,58]. It is shown that CVM leads to minimal interference among virtualized applications [59] with negligible extra energy consumption compared to those of HVM and PVM [60]. The main limitation of the CVM design is that each server is unable to run more than one host OS on the physical hardware. Because of that reason, commercial cloud providers seldom use CVM [52]. Figure 1.5 illustrates an example of CVM design with two tenants and virtualized applications for each server. Essentially, the virtualization layer installed on top of the host OS hides the complexity of the underlying system from users such that from the tenant's point of view, each application has its own dedicated OS. Some famous examples of CVM design are OpenVZ [61] and Linux-VServer [62].

1.2.1 VM Live Migration

Live migration of VM is a remarkably powerful tool, enabled by the virtualization technology. This provides the facility for cloud administrators to transfer VMs across the DCN from one machine to another with very negligible negative impact on the applications and processes being executed on the VM [63]. Dynamic re-allocation of VMs is highly beneficial for various critical system functions such as load balancing, power management, and DCN traffic management [2,64,65].

Load balancing approaches aim to balance the overall resource utilization in the data center to enhance the performance of applications running on hosts. Figure 1.6 illustrates an example of load balancing scenario. The initial CPU utilization of Host 1, Host 2, Host 3 and Host 4 are 60%, 60%, 20% and 95%, respectively (see Figure 1.6(a)). The host is over-utilized if its CPU utilization exceeds 85% of the total computing capacity of the host. Hence, Host 4 in Figure 1.6(a)) is over-utilized and unable to satisfy the computing demands of VM 5, VM 6 and VM 7. Live VM migration can effectively prevent the performance degradation of applications running

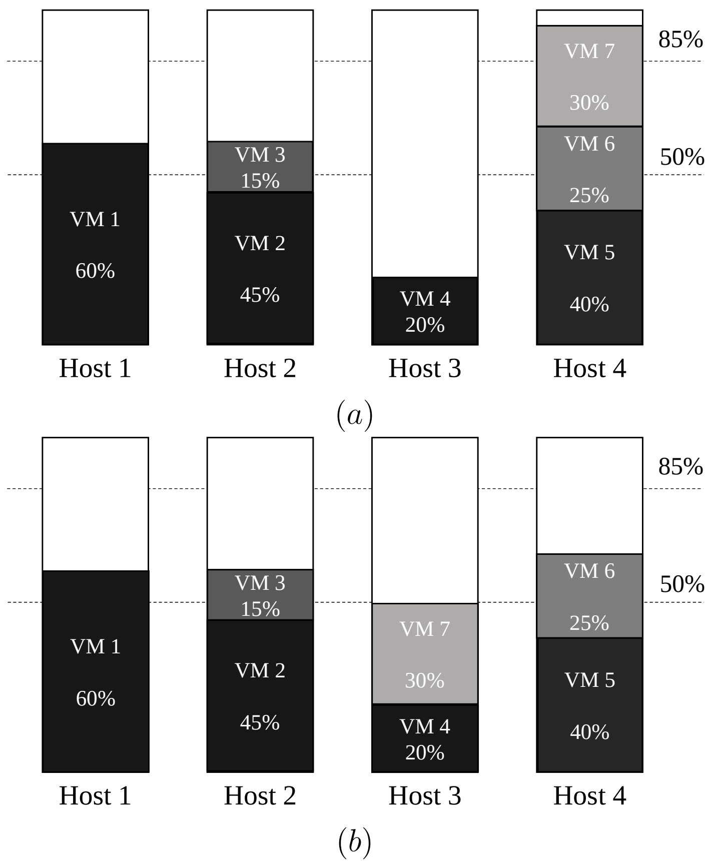


Figure 1.6: An example of load balancing VM migration (a) before migration (b) after migration [1, 2].

on Host 4 such that migration of VM 7 distributes the load evenly on the system and consequently the resource demands of VMs are met (Figure 1.6(b)).

The power management approaches address the energy inefficiency of data centers. Typically, the power management methods dynamically consolidate VMs on a minimum number of hosts. This allows operators to switch idle hosts to standby mode and eliminate the idle power consumption. However, aggressive consolidation of VMs may cause serious performance degradation of applications running on VMs. The common approach to address this issue is to migrate VMs from over-utilized hosts to less loaded hosts. Figure 1.7 depicts an example of power management VM migration. The initial VM placement in Figure 1.7(a) is similar to what is explained for Figure 1.6(a). The optimization algorithm selects VM 6 to migrate from Host 4 to Host 1. This reduces the CPU utilization of Host 4 from 95% to 65%. Moreover, VM 4 is migrated from Host 3 to Host 2 Figure 1.7(b). After migration of VM 4 is completed, Host 3 can be switched from active mode to standby mode for energy saving.

Network bandwidth is a scarce resource in data centers [66]. In particular, network intensive applications such as three-tier web applications generate huge amount of traffic on the DCN. This can quickly lead to network congestion and performance degradation of applications especially in terms of response time. The live migration of VMs is a well known approach to bring communicating VMs close to each other to minimize the network congestion and improve the application response time.

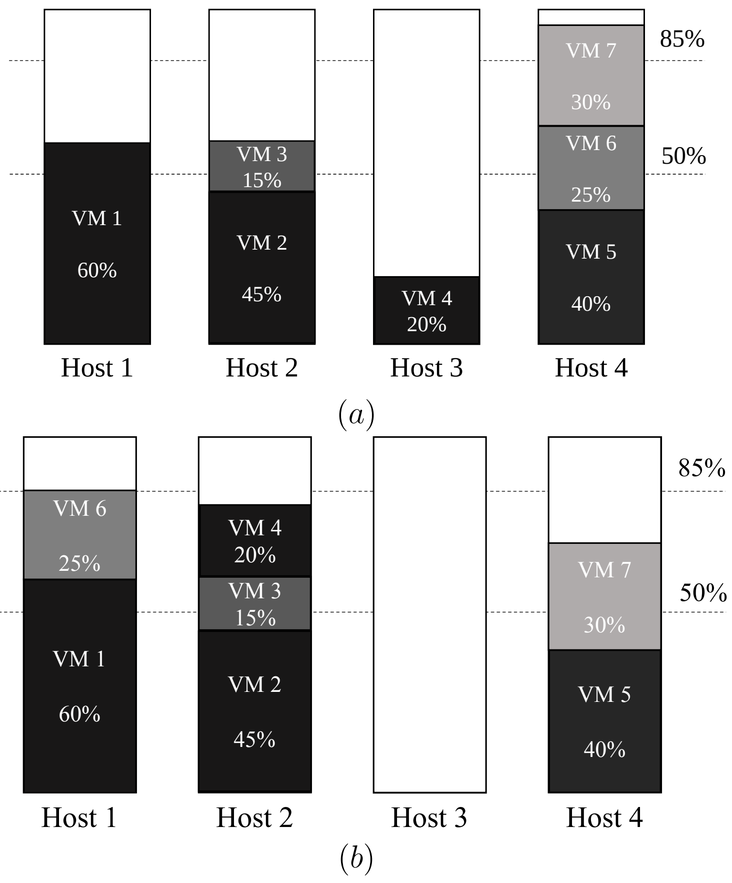


Figure 1.7: An example of power management VM migration (a) before migration (b) after migration [1, 2].

1.3 Scope, Contributions and Organization

1.3.1 Research Motivations

Typically, cloud data centers consist of powerful physical machines that are interconnected via high speed network. Cloud computing provides on-demand access to computing and network resources on a pay-as-you-go business model. This makes the Cloud an attractive solution for several types of applications such as Web service, HPC and large-scale data processing [67–69]. In fact, the ever increasing number of hosts and switches in data centers causes several issues. Firstly, such a giant data center consumes considerable amount of energy that leads to high operation cost. It is reported that the energy consumption of IT infrastructures in the United States was about 61 billion kWh that corresponds to 2 percent of global carbon emission, and the numbers are likely to double every 5 years [70]. The other issue is the high amount of traffic that is generated by applications on the data center network. This can quickly over-load the network links and apparently increase the average response time of applications. This network problem is more noticeable for network-intensive applications. Therefore, an efficient resource management algorithm is crucial for modern data centers.

There are many research works exploring the energy and network optimization of Cloud platform. However, little work has been done to study the three-tier web applications running on the cloud. Thus, the scope of this thesis is to design and implement an adaptive energy-efficient network-aware VM migration heuristic for three-tier web applications running on the Cloud platform. The main objective of the proposed approach is to minimize the SLA violation and energy consumption of the data center.

1.3.2 Organization and Contributions

The organization of this thesis is as follows. Chapter 1 provided a background of Cloud Computing coupled with the research motivation and main contributions of this dissertation. Chapter 2 discusses related works and emphasizes the novelty of contributions. Chapter 3 explains a benchmark study that is conducted to study the average response time and computational cost of three-tier web applications running on public cloud providers. Chapter 4 and Chapter 5 are the main two chapters of this work which present the design and implementation of an energy-efficient network-aware VM management algorithm for three-tier web applications running on the Cloud. In particular, Chapter 4 proposes a network-aware VM migration heuristic and determines the most suitable overloading detection policy. Chapter 5 extends the proposed network-aware approach with energy-awareness capabilities. Finally, Chapter 6 presents the conclusion and future research directions.

The organization and the main contributions of the thesis are summarized as follows.

- **Chapter 1:**
  - The fundamental background of Cloud computing is provided. Furthermore, some applications of VM migration are presented.
- **Chapter 2:**
  - The previous benchmark studies for web applications running on clouds are presented. Moreover, research works on existing energy-efficient and network-aware VM management algorithms are discussed.
  - The novelty and significance of this work is stated compared with existing research works in the literature.
- **Chapter 3:**
  - A *WPress* benchmark is proposed, which uses WordPress as a widely used open-source blogging and publishing platform in today's market [71]. WordPress follows the three-tier architecture consisting of presentation layer (Web tier), business logic layer (App tier), and database layer (DB tier). Furthermore, various use-cases for WordPress are described to indicate that it is a good representative for three-tier Web applications.
  - An open-source Benchmark Client Application, called *WPressClient*, is implemented to generate typical Web requests for WordPress and collect the results.

  - A distributed infrastructure is used to test *WPress* on small and large instance types of Amazon EC2 [37], Microsoft Azure [72] and Rackspace Cloud [73]. *WPress* runs on each cloud provider to evaluate and compare average response times and total operation cost.
  - A CPU micro-benchmark is employed as proposed in [74], to detect periods during which the VM is not assigned CPU time by the hypervisor. Results of the CPU micro-benchmark are used to investigate the effect of CPU performance on observed average response times and total operational cost.

- **Chapter 4:**
  - The high SLAV is addressed by proposing a network-aware migration heuristics that aims to minimize the SLAV of three-tier web applications running on the Cloud.Once a congested link is detected its traffic flows are monitored and the network gain is calculated for all contributing VMs. Eventually, the VM with the maximum network gain is selected for migration. The same practice applies to all congested links. The steady-state traffic condition has been used to calculate the network gain to mitigate the negative effect of VM migration on other flows. The network gain is the amount of network traffic that can be removed from the data center network after migration. The larger the network gain, the lower SLAV is expected to be achieved.
  - Two overloading detection policies are compared for the host selection policy: 1) Migration with Static Threshold (MST) that uses a static Upper Utilization Threshold (UUT), and 2) Migration with Load-balancing Threshold (MLB) in which the UUT sets to the average CPU utilization of the system.
  - Finally, to verify the effectiveness of the proposed heuristic, CloudSim version 3.0, a well-known cloud simulator, is extended with power-network awareness capabilities for three-tier web applications running on the Cloud.

- **Chapter 5:**
  - The network-aware migration heuristics proposed in Chapter 4 is extended to include power-awareness capabilities. In particular, once a congested link is identified, the algorithm monitors its traffic flows and calculates network gain and power gain values for all contributing VMs. Accordingly, the network gain and power gain lists are created for each congested link and eventually, the VM with the lowest sum of ranks is selected for migration. The power gain reflects the amount of energy saving achieved by VM migration. Two different methods are used to measure the power gain and conducted an extensive set of experiments to evaluate the performance of the proposed methods.
- **Chapter 6:**
  - The conclusion of this thesis is presented and some future research opportunities are suggested.

Chapter 2

Literature Review

In this chapter, firstly, the previous research works for benchmarking the three-tier web applications on the cloud platform are discussed. Then, the most relevant studies that addressed the energy efficiency and network optimization in the cloud data centers are presented. Finally, the research trend is explained and the novelty of thesis is presented.

2.1 Benchmark Studies on Cloud Computing

Schad et al. [75] carried out a comprehensive study on performance variation of different VM instances on Amazon EC2. A multi-node MapReduce application is used to quantify the impact of real data-intensive applications. Furthermore, they used three micro-benchmarks in their study. Firstly, the *Unix benchmark utility* (Ubench) is used for testing the CPU and Memory speed. Secondly, they employed Bonnie++ micro-benchmarks for disk performance evaluation. Thirdly, the Iperf is used to evaluate the network bandwidth. They analyzed the results in terms of different availability zones, points in time, and locations. They showed significant variation of performance in Amozon EC2.

Liang et al. [76], proposed the CARE (Cloud Architecture Runtime Evaluation) framework for evaluating different cloud application hosting servers and cloud databases and ran it on Amazon Web Services (AWS), Google App Engine and Microsoft Azure. Small compute instances on each provider were compared. It is shown that Amazon EC2 and Azure have larger response time compared to that of Google App when concurrent requests increase. In addition, different cloud databases including Amazon S3, Amazon SimpleDB, Amazon LocalDB, Azure table storage, Azure blob storage and App Engine datastore were tested in their work. This is different from my benchmark study in that Amazon EC2, Microsoft Azure, and Rackspace cloud are compared. Furthermore, the main focus of my work is on CPU utilization of cloud providers rather than their databases.

Klems et al. [77] proposed a quality measurement and analysis framework for run-time management of cloud database service systems. It can be used for analyzing the scaling strategies, and system configuration changes. This also allows more balanced database system optimization in the case of conflicting objectives. The paper proposed a parametric configuration model including three levels: 1) the compute cloud management level which is more related to capacity management, 2) the cluster management level which is mainly focused on data replication and distribution methods, and 3) server management level that is primarily concerned with operating system and software settings. The authors showed that EC2-based Cassandra cluster outperforms Amazon database services: SimpleDB and DynamoDB.

Cloud storage systems often sacrifice consistency for availability. Bermbach et al. [78] proposed a new approach to study staleness in distributed databases. The proposed approach is used to evaluate the eventual consistency in Amazon's Simple Storage Service (S3). The authors have shown that S3 frequently violates monotonic read consistency. Moreover, Bermbach et al. [79] discussed the main challenges and requirements of building a standard comprehensive benchmark that quantify the consistency guarantees of cloud hosted storage systems. The authors extended the consistency benchmark tool suggested in [78] to measure violations of Read Your Writes Consistency (RYWC) and Monotonic Write Consistency (MWC). The paper used Yahoo! Cloud Servicing Benchmark (YCSB) as a workload generator component and applied the proposed approach to evaluate the consistency guarantees of Casandra and MongoDB.

Curino et al. [80] proposed OLTP-Bench (On-Line Transactional Processing), a benchmarking approach that is designed and implemented to evaluate various relational DBMSs (Database Management Systems) and cloud-based DaaS (database-as-a-service) solutions. The proposed approach is capable of controlling the transactional mixture, rate, and workload dynamically during simulation. Different workloads are employed including YCSB [81] and Wikipedia. The experiment is run on five EC2 RDS instance sizes: Small, Large, HighMem XLarge(XL-HM), HighMem 2XLarge(2XL-HM) and HighMem 4XLarge(4XL-HM). The simulation results show that although 4XL-HM has the maximum throughput and minimum latency, XL-HM has the best price/performance ratio for the majority of customers.

The authors in [82] presented a case study in which a DaaS service provided by a DBMS with lower hourly rate costs more to clients compared to a DaaS service offered by another DBMS with higher hourly rate. The paper, proposed a new service, so-called Benchmark-as-a-Service (BaaS) that provides an accurate estimate of the actual costs charged to end users based on their workload patterns.

There are many research works that employed TPC and Rubis benchmark for the performance evaluation of web-based applications running on the cloud [83–90]. In [83], Huppler presented the important requirements of a good benchmark. In particular, they explained five key characteristics for an ideal benchmark including relevancy, repeatability, fairness, verifiability and cost effectiveness. The author believed that spending less time developing a benchmark to stress a subsystem is preferable to spend long time on designing a benchmark for the whole system. However, he argued that subsystem should not be used as a representative of the whole system. The TPC and SPEC benchmarks are examined in this paper and it is recommended to design and implement new benchmarks to address the rapid changes of the industry.

Folkerts et al. [84] extended Huppler's work [83] to present a more comprehensive classification of benchmark requirements in cloud environments. They listed sample use-cases and proposed appropriate benchmarks for each use-case including TPC-C. They also discussed meaningful metrics, variable workloads, scalability, fairness and repeatability as the main challenges for designing a good benchmark in cloud environments.

Kossmann et al. [85,86] studied the end-to-end performance and cost of executing enterprise web applications on different cloud providers. In particular, they employed

TPC-W benchmark to evaluate various database architectures offered by AWS, Google App and Microsoft Azure. The simulation results showed that in terms of cost, Google App is preferable for small applications with low workload. Nonetheless, Microsoft Azure is shown to be the most affordable cloud provider for medium and large applications.

Hill et al. [87] proposed a quantitative method in order to analyze the performance of two main services offered by Wndows Azure Platform. The first one is the Windows Azure which provides both computing resources and scalable storage and the second one is the SQL Azure that offers traditional SQL server services limited to 10 GB in size. TPC-E benchmark is used to evaluate the Azure database on three scenarios: 1) SQL-Server and clients are installed on the same machine, 2) SQL-Server and clients are located on the same local LAN and 3) SQL-Server and clients are installed on separate Azure VMs. The simulation results showed that for a single thread client the local deployment has the best performance. However, the cloud deployment resulted in better performance when the system scales up and the number of concurrent users increases.

Binning et al. [24] discussed the main reasons why TPC-W is not a suitable benchmark application for cloud computing. First, ACID properties for data opration must be guaranteed in TPC-W. However, the cloud environment typically does not offer such strong consistency constraints. Second, the main metric in TPC-W is the maximum number of web interactions per second (WIPS) that the system can handle. However, this metric is not very accurate for cloud environment in which an increasing load is ideally compensated by additional resources.

Furthermore, the second metric of TPC-W which is the ratio of cost to performance ($/WIPS) is not a good representative for the cloud cost model. This is due to the fact that the pricing in TPC/W is based on the total cost of system ownership for 3 years. Hence, it is not possible to find a single $/WIPS value for cloud environment. It is also worth mentioning that TPC is an expensive benchmark application with the annual full membership of $15000.

2.2 Resource Optimization Approaches

There has been a growing number of research works to optimize the resource allocation in cloud using live migration of VMs. This section divides previous works into three main categories: energy-aware VM migration algorithms, network-aware VM migration algorithms, and energy-efficient network-aware VM migration algorithms.

2.2.1 Energy-Aware VM Migration Algorithms

In [91], the authors proposed a methodology that can automatically detect and resolve the bottlenecks in a two-tier read-intensive web application running on the cloud. In particular, their approach is able to actively monitor the response time of requests, capture the CPU utilization of hosts, and identify the over-utilized physical machines. Once the bottleneck is detected, their proposed heuristic dynamically scales up and allocates a new server to a specific tier. The work is focused on a two-tier web applications including web-tier and application-tier. Their approach is also able to scale down for varying workload to minimize the energy consumption. They evaluated their approach with synthetic workload on a EUCALYPTUS-based cloud and showed that their method is able to guarantee the SLA.

Beloglazov et al. [92] proposed an energy-aware allocation of data center resources while meeting the SLA violation. In particular, the VM allocation problem is divided into two parts. The first part is the admission of new requests for VM provisioning. The authors employed Modified Best Fit Decreasing algorithm (MBFD). The MBFD algorithm sorts all the VMs running on the system in a decreasing order in terms of their CPU utilization and each VM is allocated to a host that results in the least increase in the total energy consumption of the system.

The second part accounts for optimization of current allocated VMs. The optimization cycle starts with selecting VMs from over-utilized hosts and then applying the MBFD algorithm to allocate them to new physical hosts. The simulation results have shown that their approach is able to save considerable energy consumption compared to static resource allocation approaches. In fact, the MBFD algorithm is used for the initial VM placement in the proposed energy-efficient network-aware approach.

In [93], an Automated Controller for Energy-aware Servers (ACES), is proposed to minimize the energy consumption of a data center while the workload demand is met. The ACES starts with characterizing the incoming workload and predicting the workload demand in near future. Then, the server power scheme is modeled as a Markov state diagram in which each state represents a server power scheme (e.g. on, off, sleep, hibernate), and each edge denotes the cost of state transition. The cost is calculated based on the transition delay, the cost of energy consumption, and reliability of the server. The problem is approximated through a linear programming formulation and solved by the polynomial-time approach proposed in [94]. The simulation results revealed that the ACES can achieve considerable energy saving while meeting the workload demand by identifying the optimal assignment of power schemes to servers.

Younge et al. [95] proposed a new energy efficient framework for cloud computing architecture. They suggested a greedy-based VM scheduling algorithm to minimize the energy consumption of a data center. For the given list of new VMs, the algorithm checks the number of required CPU cores for each VM. Their approach is an effort to allocate as many VMs as possible to each physical host. Moreover, a simple VM management approach is discussed. Live VM migration is applied to move VMs running on low load hosts to medium load servers. Once all VMs are migrated from a low load host it is switched off to save the energy consumption.

In [96, 97], authors proposed a new scalable and energy efficient VM management framework for a private cloud so-called Snooz. The Snooz has a hierarchical structure consisting of four layers. The bottom most layer is known as Local Controller (LC) which is connected to physical hosts. The next layer is called Group Manager (GM) such that each GM manages a subset of LCs in terms of VM monitoring, resource utilization, VM scheduling, and power management. The third layer is Group Leader (GL) that receives requests from clients and dispatches them among GMs. In fact, the energy-aware VM scheduling is performed in two steps: from GL to GM, and from GM to LC.

Their proposed approach used round-robin method to dispatch new VM requests from GL to GMs. The GMs used round-robin and first-fit placement policies to assign new VMs to LCs. The VM migration is performed in GM level in the event of LC over-loading or under-loading. The paper migrates the VMs from over-loaded LCs to lightly-loaded LCs. Furthermore, the energy saving is achieved by migrating all the VMs from low-loaded LCs to moderately-loaded LCs.

Kaur et al. [98] proposed an energy-efficient approach for cloud computing using genetic algorithm. The SLA parameters considered in their work are the system response time and throughput. The common VM management strategy is employed. Firstly, the over-loaded and under-loaded servers are detected. Secondly, the MBFD [92] algorithm is used for initial VM placement. Finally, the VM selection policy selects VMs for migration. The VM migration is also based on lower and upper threshold values of the physical hosts. The simulation results showed that their approach can reduce the energy consumption of the data center while improving the response time and throughput of the system.

In another work [99], an energy-efficient VM placement approach was proposed based on Minimum Correlation Coefficient (MCC) method. The authors used Fuzzy Analytic Hierarchy Process (AHP) to make a suitable trade-off between energy consumption and SLA violation of a data center. In particular, the paper modeled the VM placement algorithm as a bin packing problem with variable bin sizes and prices. The VM migration is selected based on two criteria.

First, the MBFD algorithm [92] is used such that the migrant VM leads to the least increase in the energy consumption of the data center. Second, the MCC method is employed such that migrant VM has the minimum correlation between existing VMs on the destination host in order to reduce the SLA violation. In fact, the high correlation between resource usage of VMs running on a host denotes a high risk of host over-utilization and SLA violation. The experimental results have shown that the proposed approach can provide a suitable trade-off between energy consumption and SLA violation in the cloud.

Monil et al. [100] proposed an energy-aware VM consolidation approach known as Maximum Correlation with Migration Control (MCMC). Their work is mainly focused on the VM selection policy that is applied once over-utilized hosts are detected. In particular, they calculated the correlation between all VMs on the over-utilized host and selected the VM with the maximum correlation with other VMs. The main reason for this selection is that the VM with the maximum correlation with other VMs has a high probability for server over-loading. Furthermore, the paper considered another constraint called migration control.

The main idea of migration control is to prevent migration of VMs with steady CPU usage. This is due to the fact that such VMs may over-utilize any hosts in the data center. Therefore, they suggested that the CPU utilization of the selected VM should not exceed a predefined CPU utilization threshold. The efficiency of MCMC VM selection policy is compared with the Maximum Correlation (MC) policy in CloudSim. Five overloading detection policies are studied: Static CPU Utilization threshold (THR), Adaptive Median Absolute Deviation (MAD), Adaptive Interquartile Range (IQR), Local Regression (LR), and Robust Local Regression (RLR). The simulation results have proved that MCMC is able to improve the energy consumption of the data center in most cases.

In [101], the existing energy efficient VM management algorithms are divided into three categories including energy-aware VM allocation, energy-aware VM migration, and energy-aware task scheduling. Their work is an effort to address the first two categories. In particular, they proposed HGACSO, an energy-aware VM allocation algorithm based on Genetic Algorithm (GA) and Cat Swarm Optimization (CSO). Initial chromosomes are used as cats in the CSO cycle. Each iteration of CSO evaluates the fitness of cats and saves the best cat positions. After that, it determines whether the cat is in the seeking mode or tracing mode. Then, the algorithm applies crossover and mutation operations of GA. Next step in CSO cycle, is to select the fittest cat and chromosomes. This cycle continues until the best physical machines are identified for new VMs.

Moreover, the authors proposed an energy saving method for under-utilized hosts. Once an under-utilized host is detected, all of its VMs are migrated to other hosts using the First Fit Decreasing (FFD) algorithm. In fact, the FFD algorithm, sorts available physical machines based on their CPU utilization in decreasing order and migrates VMs using First Fit allocation method. The authors compared four allocation algorithms: HGACSO, GA, CSO, and First Fit. The simulation results have shown that HGACSO is able to save more energy consumption compared to other algorithms.

In [102], the authors proposed a Topology-Aware Virtual Machine Selection (TAVMS) algorithm for balancing the load among distributed data centers. The main objective of this work is to reduce the total energy consumption of a system. They assumed a single cloud provider with several data centers around the world, which are connected via a 100 Gbps backbone optical network. Incoming requests from clients are modeled as a group of VMs communicating with each other. Moreover, it is assumed that VMs have traffic flows to the Internet. The TAVMS operation includes two phases. In the first phase, the Selection (known as SEL) algorithm, periodically evaluated each data center and identified the potential VM groups for migration.

In the second phase, the Negotiation (NEG) algorithm estimated the energy consumption of the VM groups after migration to other data centers and selected the data center with the maximum energy saving. For each VM group, the NEG algorithm estimated the migration time based on the disk size, size of RAM, and the available bandwidth. The algorithm prevents migration of VM group in which remaining lifetime of VMs is larger than the migration time. The simulation results have shown that TAVMS resulted in significant saving in energy consumption compared to scenarios with no migrations.

Wang et al. [103] proposed a decentralized Multi-Agent-based (MA-Based) resource management method for cloud computing systems. In particular, a cooperative agent is assigned to each physical machine in order to control the resource management on the physical machine. Upon receiving a set of VM requests from clients the MA-based approach run two procedures to allocate VMs on hosts.

Firstly, an auction-based VM allocation algorithm is used for initial VM allocation. The agents and VMs are modeled as bidders and commodities, receptively. At each bidding cycle, the agent bids for the largest VM that it can host and it broadcasts its bids to all other agents in the system. Once the broadcasting is completed all the agents send an acknowledgment message to the winner agent which is the one that can host the VM with the minimum energy consumption. Secondly, a local negotiation-based VM consolidation mechanism is employed such that agents use VM live migration to exchange their allocated VMs with the aim of minimizing the energy consumption.

2.2.2 Network-Aware VM Migration Algorithms

In [104] an efficient network-aware resource management algorithm is proposed for distributed data centers. The authors assumed multiple small data centers which are interconnected over a wide-area network. This assumption is different from typical resource management systems that consider a few number of large data centers. Each request from client specifies the number of VMs required to be provisioned. The major objective of this work is to minimize the maximum latency in communication between VMs that are allocated for each user. In particular, the paper used the 2-approximation algorithm to find the optimal data centers for the VMs. Also the same algorithm is used to find the optimal physical machine inside the data center. The simulation results have shown the efficient performance of the proposed algorithm.

In another work, Biran et al. [105] introduced a stable network-aware VM placement mechanism for cloud computing. Their proposed VM placement approach, known as Min Cut Ratio-aware VM Placement (MCRVMP), is an effort to satisfy the predicted communication demand considering time-varying nature of the network traffic. The MCRVMP modeled data center network as a graph and used the concept of network graph cut in order to reduce the worst case cut ratio. A general graph may have an exponential number of cuts.

Therefore, the paper only considered critical cuts which are bottlenecks for the traffic between VMs that are placed in opposite sides of the cuts. Eventually, the

MCRVMP identified the VM-Host placement that minimizes the maximum network cut load, while satisfying CPU and memory demands. The simulation results have shown that their approach can reduce the number of packet dropping and VM relocations.

In [106], authors considered a network-aware mechanism for coordinating concurrent VM migrations in cloud environment. The paper used a simulation tool for predicting the total completion time of concurrent VM migrations based on different VM characteristics and network links conditions. The pre-copy VM migration technique is used in the paper and it is shown that available bandwidth on the network link and the dirty page rate of the migrating VM are the essential factors for finding the optimal link sharing strategy.

An extensive simulation study has been conducted to obtain the optimum number of parallel migrations for the given network bandwidth and dirty page rate. The simulation results have shown that the total completion time of concurrent VM migrations is lower compared to that of sequential VM migration for low dirty page rate and high network link bandwidth. Otherwise, the sequential VM migration is preferable.

Concurrent VM migration is also studied by Chen et al. [107]. The authors proposed a network-aware load-balancing approach using concurrent VM migrations. The paper is an attempt to minimize the time required for load balancing. The load balancing problem is modeled as a minimum weighted matching problem and is solved using Hungarian method [108]. In fact, the over-utilized and under-utilized are separated in two different sets of vertices known as trigger nodes and non-trigger nodes, respectively. The edge weight between vertices is determined by resource demand of VM, resource capacity of physical machines, and the network cost.

The network cost is modeled as the number of hobs between two servers in the data center network. The algorithm identifies a set of VMs from trigger nodes and migrates them to non-trigger hosts with the objective of reducing the total migration time. The experiments are conducted in NetworkCloudSim [109] and 100 Mbps network bandwidth was reserved for concurrent VM migration. The experimental results have shown that the proposed approach is able to minimize the time required for load balancing.

Network flows typically traverses through a sequence of policies which are specified by middleboxes such as firewall, load balancer, traffic shapers, etc. Cui et al. [110] proposed a PoLicy-Aware Network-aware, known as PLAN, VM migration algorithm for clouds. The main objective of PLAN is to minimize the communication cost while the policy requirements are met. The communication cost is specified in terms of the policy requirement of VMs. The PLAN algorithm is a distributed heuristic approach executed on both VM and server sides known as PLAN-VM and PLAN-server, respectively.

The optimization process starts from PLAN-VM which employs a greedy approach to find the most appropriate destination host for each VM. The results are sent to PLAN-server requesting for final approval. The request will be approved by PLAN-server provided that the resource demand of VM does not exceed the available resources on the destination host. Upon approval, the migration process starts. The proposed approach has been evaluated in ns-3 [111] using a fat-tree topology. The simulation results have shown that PLAN is able to minimize the communication cost while the policy constraints of network flows are satisfied.

Vijay et al. [65] proposed Remedy, a network-aware steady-state VM migration approach for data centers. The main objective of Remedy is to reduce the network congestion from the data center while minimizing the migration cost. The migration cost is defined as the total network traffic that is generated during migration. Remedy identified the candidate flows that pass through congested links. It employed a heuristic-based approach to remove network congestion.

At the very beginning, the VM selector algorithm calculates the migration cost associated with VMs that are part of the candidate flows. After that, the target selection algorithm monitors resource usage of physical machines and ranks them based on available bandwidth and bandwidth balance after migration. Remedy has been evaluated in VMFlow [112] using real network traffic data. The simulation results have shown that Remedy is able to predict the migration cost with high accuracy.

Moreover, it is shown that Remedy can efficiently exploit unused network bandwidth in order to reduce the network congestion in the data center.

Piao et al. [113] suggested a network-aware VM placement and migration approach for data-intensive applications running on a cloud. The paper assumed a federated cloud system consisting of several compute and storage clouds that are interconnected via Internet or Intranet links. Clients send compute and storage requests to the cloud provider. The data center broker allocates requested VMs and storage blocks to the compute cloud and storage cloud, respectively. In fact, clients use VMs to run their applications while the real data is stored in the storage block.

The authors suggested a VM placement policy that chooses a physical machine that results in the smallest data transmission time between application and storage blocks. Nonetheless, communication delay is a typical challenge for data-intensive applications that can be caused by network congestion or faulty links. The authors addressed this challenge by proposing a network-aware VM migration algorithm that identifies a destination host such that the transmission time between applications and storage blocks remain minimized. The simulation results in CloudSim 2.0 have shown that the proposed placement approach is able to improve the average task completion time, despite the fact that the migration algorithm is not very effective.

Stage et al. [114] is the vision paper to propose a network-aware topology-aware model for scheduling VM migration in clouds. The paper discussed the optimization of network bandwidth consumption due to VM migration. The proposed model consists of four major units including workload classifier, allocation planner, live migration scheduler, non-conformance detector. The workload classifier is designed to identify different class of workload and allocate workload on one class in the same server cluster. The workload classifier considered several attributes to identify the class of workloads including predictability and periodicity.

Once workload is classified, the allocation planner determines the expected over-utilized and under-utilized physical machines in the data center and finds the suitable VMs for migration. The non-conformance detector is provided for handling spike in resource utilization in order to prevent over-utilization of physical machines. Finally, the live migration scheduler gets the list of VMs for migration from allocation planner and non-conformance detector units and calculates the optimal time to start the migration. Although the authors did not propose an actual algorithm and any simulation results, their proposed model can be useful to minimize the SLA violation due to live VM migration in data centers.

Another work is a VMbuddies done by Lio et al [115]. The VMbuddies is a coordinating approach for live migration of three-tier web application across distributed data centers. The authors addressed the correlated VM migrations problem by proposing an adaptive bandwidth allocation approach to minimize migration cost. The migration cost is defined in terms of migration completion time and migration downtime. Similar to my work, it is assumed that the application traffic and migration traffic shared the same bandwidth.

However, the available bandwidth for migration between clouds is much smaller compared to the bandwidth inside a data center. Hence, the VMbuddies methodology is an effort to coordinate VM migrations such that the bandwidth between clouds can be exclusively used by the migration traffic. In particular, all the VMs of a web application are identified and migrated into the same data center. Moreover, a synchronization approach is employed such that VM migrations are completed simultaneously. This is to avoid retransmission of data across data centers.

The authors used RUBBoS [116], a public three-tier application benchmark, to evaluate their proposed approach. The algorithm is tested on both simulated Wide Area Network (WAN) and real in-built private clouds between Nanyang Technological University (NTU) and National University of Singapore (NUS). The experimental results have shown that VMbuddies is able to make substantial improvement in the performance degradation experienced by three-tier application during migration with slight overhead due to migration synchronization.

2.2.3 Energy-Efficient Network-aware VM Migration Algorithms

Vu et al. [117] suggested a virtual machine placement approach that takes energy consumption and network traffic into consideration. In particular, their approach is very similar to conventional VM migration algorithms consisting of the following phases. The first phase detects the over- and under-utilized hosts. The second phase selects the VM for migration. One or more VMs are selected from over-utilized hosts such that the CPU utilization falls below the over-utilization threshold.

Furthermore, energy saving is achieved by migrating all VMs from under-utilized hosts and switching them from active mode to standby mode. Finally, the third phase employs a network-aware target selection policy for the selected VMs such that those with high communication traffic are placed close to each other. However, this approach is not very efficient for network-intensive applications with small CPU demand. The simulation results in CloudSim have shown the efficiency of the proposed approach.

Another work is that of Kliazovich et al. [118]. They presented DENS as an energy-efficient network-aware scheduling methodology for data centers. Their approach is designed to avoid hotspots in data center network while minimizing the number of hosts for job scheduling. The paper also highlighted the importance of communication fabrics in the total energy consumption of data centers. It suggested to keep the core switches running while turning off the aggregation switches whose racks are inactive. In particular, the DENS algorithm is an effort to minimize the energy consumption of a data center by finding the best server for executing a given job.

Each iteration of the DENS algorithm calculates several factors including server load, rack load, and communication potential of racks. Then, the algorithm computes the DENS metric which is a weighted sum of the calculated factors and finally selects the server with the maximum DENS metric. The communication potential is defined as the total amount of bandwidth provided to each server in the data center. The DENS methodology is evaluated on GreenCloud simulator [119]. The simulation results have shown that DENS methodology is able to optimize the tradeoff between job consolidation and the traffic pattern of a data center.

In [120], a network- and power-aware, so-called NAP, consolidation approach is introduced. The main objective of the NPA algorithm is to identify an appropriate migration map with the least number of VM migrations such that power consumption is reduced and network performance is improved. The NPA consists of two main modules. The first one, is the power module that employed the Threshold with Random Selection (THR-RS) algorithm proposed in [121]. The THR-RS uses static threshold to detect over- and under-utilized physical machines. Furthermore, it employs the random selection policy for VM selection from over- and under-utilized hosts. Readers are referred to [121] for more details about THR-RS algorithm.

The second one, is the network module that is specifically designed to address network overhead imposed by VM migrations. In fact, the optimization cycle is triggered in three conditions: 1) host over-utilization, 2) host under-utilization, and 3) low throughput between communicating VMs. The paper assumed a predefined throughput below of which is considered as low throughput. The destination selection policy depends on how the optimization is triggered. If migration is triggered due to over- and under-utilized hosts, the algorithm only selects a host that meets the computing demand of VM regardless of its impact on the network traffic. Otherwise, it tries to place the VM on hosts near each other ensuring resource demand is met. The author used CloudSim to compare THR-RS, NPA, and DVFS [121] approaches.

Not surprisingly, the simulation results have shown that NPA has lower SLA violation and higher throughput compared to that of THR-RS. However, the DVFS has significantly lesser SLA violation and better network throughput compared to NPA. The author believes that the low efficiency of NAP approach is attributed to two facts. First, is the random VM selection policy. Each VM might communicate with different number of VMs with varied traffic patterns. Thus, random selection of VMs may quickly degrade the system throughput and increase the SLA violation and power consumption. Second, is ignoring the negative effect of VMs migrated from over- and under-utilized host on the network traffic. This may lead to negative consequences especially in terms of network traffic for network-intensive applications.

Lin et al. [122] considered integration of routing algorithms and VM migrations in cloud data centers. The main objective of the paper is to optimize the network throughput while the total energy consumption of data center is maintained. The authors used a cloud architecture consisting of four major parts: 1) network devices,

2) Software Defined Network (SDN) [123] controller that is in charge of managing the network devices, 3) computing devices, 3) cloud controller which is working as a hypervisor and responsible for VM management and resource allocation. In particular, the paper contributions are two folds.

Firstly, the authors suggested a dynamic reroute with flow migration (DENDIST-FM) aimed at improving the network performance and avoiding network congestion. The DENDIST-FM algorithm continuously monitored the communication links. Once, an over-utilized link is detected the algorithm collects flow information from the link, re-computes new path and updates new route.

Secondly, they proposed an energy-and-topology aware VM migration (ETA-VMM) algorithm to determine the optimal migration map. The algorithm employed minimum migration technique [121] to select VMs from over-utilized host and add them to list of candidate VMs for migration. Moreover, all VMs from under-utilized hosts are selected and added to the list of candidate VMs for migration. Finally, the algorithm identifies destination host based on the network distance which is modeled as the number of hops.

The authors used Network Simulator (NS2) [124] and Cloudsim v3.0 to evaluate their approach. NS2 is used to simulate source routing and flow-aware routing in SDN and evaluating the network throughput. CloudSim is used to simulate the VM management algorithm and evaluate the total energy consumption. The simulation results have shown that the proposed approach achieves throughput improvement and energy saving.

Reguri et al. [125] is another network-aware approach that aims at reducing energy consumption of a data center while SLA is met. The authors employed three energy-efficient approaches in [126] which are enhanced versions of the MBFD algorithm [92]. The first approach is the Least Increased Power with host sort (LIPHostSort) algorithm that finds all the destination hosts with the least increase in power consumption, sorts them in decreasing order of their CPU utilization, and selects the one with the highest CPU utilization. The second approach is the Best Fit Host (BFH) algorithm.

The main idea is to select the host whose predicted CPU utilization after VM migration is maximized without exceeding the upper utilization threshold. The last approach, is the Best Fit VMs (BFV) algorithm that employs dynamic programming approach to find the best-fit VM for each host.

The authors proposed three energy-efficient traffic-aware VM management algorithms by integrating the traffic-aware VM Clustering (VC) approach presented in [117] with any of the three aforementioned algorithms. The new algorithms are known as LIPHostSort with VC (LIPHostSort_VC), BFH with VC (BFH_VC), and BFV with VC (BFV_VC), correspondingly. Once an over-utilized host is detected, the traffic-aware approach clusters the communicating VMs and looks for the destination to migrate the whole cluster to another host. In the same way, the algorithm identifies the VM clusters from under-utilized hosts and migrates the whole cluster to achieve energy saving. The paper used CloudSim to evaluate the efficiency of proposed algorithms. The simulation results have shown that BFV_VC leads to lower SLA violation and energy consumption compared to the other two algorithms.

Another work that deals with the network and energy efficiency in data centers is the one by Takouna et al. [127]. The authors have proposed a network-aware and energy-efficient scheduling approach for parallel applications running in the virtualized data centers. The main objective of the paper is to reduce the network utilization and improve the SLA violation. Moreover, the paper aims to minimize the total energy consumption of servers and switches. In particular, the paper proposed Peer VM Aggregation (PVA) algorithm that aggregates communicative VMs on the same host. For each VM the algorithm receives the list of all communicating VMs and tries to converge the whole system. The converged system is defined as a system on which all VMs for the same application are located on the same host.

The PVA algorithm consists of four steps. Firstly, the given list is sorted in decreasing order of the number of in/out traffic flows and the current placement of peered VMs is determined. Next, is to select the VM with the highest number of in/out traffic flows and the most appropriate destination host in terms of network traffic. After that, the Migration Manager (MM) approves the destination host provided that it meets the resource demand of migrating VM. Finally, the MM schedules the migration.

In the third step, if the MM realizes that the destination host does not have enough space for the migrating VM, it recessively calls the PVA algorithm to migrate one VM from destination host. In fact, this policy seems not to be very efficient for network-intensive application because it increases the number of VM migrations which may cause significant overhead on the data center network. The authors used CloudSim and NAS Parallel Benchmark (NPB) [128] to compare their proposed approach with a typical CPU based placement algorithm suggested in [129]. Not surprisingly, the simulation results have shown the lower SLA violation and energy consumption of PVA algorithm compared to that of CPU based placement algorithm.

2.3 Research Trends, Scope and Novelty of the Thesis

The literature review shows that different criteria are employed to evaluate the cloud platforms. The main criteria include response time, CPU speed, RAM speed, storage performance, and total price. Furthermore, several cloud providers have been studied consisting of Amazon EC2, Microsoft Azure, and Google App. Moreover, various workloads have been used such as YCSB, Wikipedia, TPC-C, RUBiS, and TPC-W. The most relevant workload type to our proposed approach in Chapter 3 is TPC-W which is designed for web applications. However, Binning et al. [24] explained the main reasons why TPC-W is not a suitable benchmark application for cloud computing. Chapter 3 is my attempt to address the main issues of using the TPC-W workload on cloud environment. In terms of cloud providers, the previous research works rarely compared large instances of the aforementioned cloud providers. Therefore, Chapter 3 compares small and large instances of Amazon EC2, Microsoft Azure, and Rackspace Cloud. Regarding the performance criteria, this thesis adopts the response time and total cost as two of the most important performance metrics for evaluation of three-tier web applications. Furthermore, a CPU micro-benchmark is employed to compare the CPU speed of the three cloud providers.

The proposed benchmark application in Chapter 3, so-called *WPress*, is different from existing benchmarks, as it uses the real-life WordPress blogging and publishing platform. The author argue that this approach has several advantages over the simulated E-commerce applications used in TPC and RUBiS. First, WordPress was ranked first among the 15 best blogging and publishing platforms on the Internet [130]. Second, it is based on the latest technologies used for web applications. The current stable version, at the time of writing this thesis, is 4.7 released on November 2016. Third, it is open-source and has a remarkable user friendly interface, which meets benchmark's availability requirement discussed in [84]. Fourth, it is reported that almost 8% of all websites on the Internet are based on WordPress. This is because more than 10000 plugins add almost all kinds of functionalities for a three-tier web application such as social media sharing, search engine optimization (SEO), photo slide shows, and on-line shopping [131]. Last but not the least, *WPress* measures total VM cost, which is a meaningful price metric to assess public cloud providers with the pay-as-you-go business model.

Section 2.2 reviews some of the previous literatures on the resource optimization approaches in cloud computing. The previous literatures can be classified into three main categories: 1) energy-aware methods, 2) network-aware approaches, and 3) energy-efficient and network-aware algorithms. The energy-aware VM migration algorithms aim at minimizing the total energy consumption of data centers. The research works were mainly aimed at minimizing the number of over-utilized hosts and switching-off as many under-utilized hosts as possible. Different approaches including scaling up/down, MBFD, FFD, greedy-based, and genetic algorithms were employed either for admission of new VMs or VM migration. The migration algorithms normally start once a new over-utilized or under-utilized host is detected in the data center. Furthermore, several criteria are considered as SLA such as response time and throughput.

The network-aware methods were mainly focused to remove as much traffic as possible from the network links. Several approaches are proposed such as 2-approximation algorithm, Min Cut Ratio-aware VM placement algorithm, greedy approach, removing network congestions from the data center network, and adaptive bandwidth allocation approach. Furthermore, algorithms were evaluated by simulation or on real data centers.

The energy-efficient network-aware algorithms normally consider both energy consumption and network traffic for VM management. Several approaches were suggested including: considering the communication fabric, turning off the aggregation switches whose racks are inactive, threshold with random selection policy, and DVFS. Proposed approaches are evaluated either by simulation such as NS2, CloudSim, GreenCloud, and SDN, or through implementation on real data centers.

This dissertation proposes an energy-efficient network-aware VM migration algorithm. The MBFD algorithm is used for the initial VM placement algorithm, and the average response time is measured as the main SLA criterion. The algorithm is evaluated through an extensive set of experiments in CloudSim. Moreover, CloudSim version 3.1 is extended to enhance its capability to simulate combined energy and network aware VM allocation algorithms for three-tier web applications. The main reason to focus on three-tier web applications is to be consistent with the benchmark study presented in Chapter 3. In fact, web-applications are small in size but very large in quantity. Thus, designing an accurate network-aware resource management technique has a great deal of importance.

The proposed approach in Chapter 4 employs the energy-aware MBFD algorithm [92] for the initial VM placement. Furthermore, the system monitors the data center network and starts the optimization algorithm as soon as a new congestion is detected in the data center network. Essentially, the VMs whose migration can remove the largest steady-state network traffic from the data center network are selected for migration. The suggested network-aware algorithm is very similar to the Remedy [65]. My work is different from [65] in that the MBFD algorithm [92] is employed to optimize the energy efficiency of the data center. Moreover, a three-tier model is designed for web applications in CloudSim and the algorithm is customized accordingly.

Chapter 5 extends the network-aware migration heuristics presented in Chapter 4 with energy-awareness capabilities to make farther improvements in total energy consumption of the data center. In particular, once a congested link is identified, the algorithm monitors its traffic flows and calculates network gain and power gain values for all contributing VMs. Accordingly, the network gain list and power gain list are created for each congested link and eventually, the VM with the lowest sum of ranks is selected for migration.

The proposed approach in Chapter 5 is different from the aforementioned studies in Section 2.2.3 in several ways. Firstly, my proposed approach considered the steady-state amount of traffic which is more accurate than considering the current traffic condition. Typically, a VM is in communication with many other VMs and migration of one VM affects the whole data center network. while a VM migration can remove network traffic from one congested link, it can imposes network traffic on other links and make them congested. Hence, looking at the steady-state amount of traffic can guarantee the optimal selection of VM for migration. Secondly, the optimization cycle in my suggested approach starts once a congested link is detected. This is a more efficient method rather than starting the optimization cycle from over- or under-utilized hosts [117,120,125]. This is because of the reason that each optimization cycle imposes some computing overhead into the system. Thus, it makes more sense to leave the system work normally with its maximum amount of computing power while there is no network congestion in the data center network. Thirdly, the proposed approach selects VMs that can remove the largest amount of traffic from the communication links. This is different from selecting VMs from over- and under-utilized hosts and applying a network-aware destination selection policy [117,122,125]. The approach is more promising than the previous methods because network traffic plays a significant role on the performance of network intensive applications.

Chapter 3

WPress: A Performance Benchmark For Three-Tier Web Applications Running on the Cloud

Cloud computing [132] has become an attractive platform for three-tier Web applications. Recently public Cloud platforms are employed in preference to specialized clusters and supercomputers due to their reliability, scalability and cost effectiveness. The Infrastructure-as-a-service (IaaS) model [9] enables customers to rent computing and storage resources from different cloud providers in the form of VMs.

TPC-W and RUBiS benchmarks have been widely used to evaluate the performance of three-tier Web applications in clouds [83, 85–87]. Nevertheless, they are not originally designed for cloud computing platforms with unknown hardware configuration and pay-as-you-go pricing model [24, 133, 134].

To address this research gap, this chapter introduces a performance benchmark for three-tier Web applications deployed on public clouds. In fact, design and implementation of a comprehensive performance benchmark in a highly distributed computing environment raise considerable challenges. First, it is a time consuming task which involves several repetitions to present statistically reliable results. In addition, choosing a representative application and generating enough load need to be carried out proficiently. Furthermore, considerable amount of data generated from long-running experiments requires a suitable data collection approach. Moreover, basic requirements of benchmarks like fairness, relevancy, verifiability and cost effectiveness, as already discussed by Folkerts and Huppler [83, 84], should be taken into consideration. In summary, the following contributions are made to address the challenges mentioned above:

- A *WPress* benchmark is proposed, which uses WordPress as a widely used open-source blogging and publishing platform in today's market [71]. WordPress follows the three-tier architecture consisting of presentation layer (Web tier), business logic layer (App tier), and database layer (DB tier). Furthermore, various use-cases for WordPress are described to indicate that it is a good representative for three-tier Web applications.

- An open-source Benchmark Client Application, called *WPressClient*, is implemented to generate typical Web requests for WordPress and collect the results.

- A distributed infrastructure is used to test *WPress* on small and large instance types of Amazon EC2 [37], Microsoft Azure [72] and Rackspace Cloud [73]. *WPress* runs on each cloud provider to evaluate and compare average response times and total operation cost.

- A CPU micro-benchmark is employed as proposed in [74], to detect periods during which the VM is not assigned CPU time by the hypervisor. Results of the CPU micro-benchmark are used to investigate the effect of CPU performance on observed average response times and total operational cost.

The rest of this chapter is organized as follows: Section 3.1 describes design and implementation of *WPress*. Section 3.2 illustrates and discusses the results of the proposed benchmark. Finally, Section 3.3 summarizes the chapter.

3.1 Methodology

This section introduces the proposed benchmark, called *WPress*, to study the performance of cloud providers when they host three-tier web applications. Firstly, WordPress is described as the benchmark application and its advantages over E-commerce applications used in TPC and RUBiS are explained. After that, the functionality and implementation details of *WPress* client application, called *WPressClient*, are described. This section ends with explanation of the experimental environment developed to run *WPress* on Amazon EC2, Microsoft Azure and RackSpace cloud. Please note that all information provided in this chapter is based on my case study research from 01 October 2013 to 30 February 2014.

3.1.1 WordPress-Based Benchmark Application

WPress is designed based on WordPress blogging and publishing platform identified as the best blogging and publishing platform on the Internet [135]. WordPress can be used for several transactional use-cases such as E-commerce, on-line booking systems and blogging platforms. Moreover, WordPress is a web application that relies on three tier architecture [136]. PHP and HTML are used in presentation tier, PHP is used as application tier, and MySQL is employed as database tier. In *WPress*, the blogging platform is used because: (1) the author believes that popularity of blogging systems is increasingly growing, and (2) WordPress has more than 25000 plugins and 10000 themes which make it the first solution for most blog owners [71].

There are a plethora of free, or at least cheap, plugins available for WordPress that can be incorporated with *WPress* in the future. For instance, WordPress provides a full fledged E-commerce website with integrated shopping card flow and PayPal payment gateway. Available plugins include *WP E-commerce* [1], *eShop* [2], *WooCommerce* [3] and *Quick Shop* [4]. Moreover, it can also be employed as an on-line booking system with *Rezgo Online Booking* [5] and *EzyOnlineBooking* [6] plugins. Other use-cases for WordPress include Job board or classified website, private social network, portfolio as well as news websites [137].

Extensibility, representativeness, availability and cost effectiveness are the main requirements for a good benchmark application [84]. The first two requirements are

[1]https://wordpress.org/plugins/wp-e-commerce/
[2]https://wordpress.org/plugins/eshop/
[3]http://wordpress.org/plugins/woocommerce/
[4]https://wordpress.org/plugins/quick-shop/
[5]https://wordpress.org/plugins/rezgo-online-booking/
[6]https://wordpress.org/plugins/ezyonlinebookings-online-booking-system/installation/

guaranteed because WordPress is implemented by the latest Web technologies and various plugins can be easily associated with its main core. Therefore, it can be extended to a benchmark suite comprising several Web services. In TPC and RUBiS, such flexibility does not exist because they are specifically designed for e-commerce applications. Availability and cost-effectiveness are also guaranteed by wordPress. This is because of the facts that: 1) WordPress is an open-source project with many available plugins, and 2) It has a user friendly interface which makes it easy to be used even by unskilled clients.

3.1.2 Client Implementation

WPressClient is the integrated client application implemented for *WPress*. It is an open-source multi-threaded Java application developed using Selenium Web-driver library [138]. Selenium Web-driver uses Firefox plugin to steer an actual browser window, so that execution of each thread in the *WPressClient*, is as close to a real user who works with WordPress as possible. Essentially, *WPressClient* has two functionalities in this chapter: (1) to generate representative workloads for WordPress and (2) to calculate average response times and total cost of VMs on the clouds under test.

Workloads are generated based on ten predefined WordPress tasks named task 1 to task 10 (Table 3.1). Each entry of the Table includes a short description of what each task does and the maximum number of HTTP requests sent to WordPress server. Each task is associated with realistic delay and scrolling actions, to mimic the real behavior of a human user of the system. Task 1 is designed to search a random keyword. It consists of three Web requests. The first request opens WordPress landing page, which is then scrolled down for 1000 pixels (which represents the user is trying to view the whole page). Afterwards, *WPressClient* waits for 6 seconds (a quick view of less than 10 seconds is assumed experimentally) before scrolling up again. Within 12 seconds (which is experimentally determined and corresponds to the elapsed time for a user to choose a keyword for search) a sample keyword (a short keyword is assumed experimentally to have ten-characters long) is typed into the search box. In six seconds (which is the experimentally determined estimated time for typing the

Table 3.1: List of tasks generated by *WPressClient*

Task	Description	Number of Web requests
1	Searching a keyword	3
2	Publishing a new post	5
3	Browsing several pages	8
4	Replying to a post	4
5	Loading a page	5
6	Uploading several photos	5
7	Deleting some posts	7
8	Creating a new post with draft status	5
9	Adding a new user	6
10	Approving several comments on pending status	6

keyword) a search request is sent. Resulting page is scrolled down for 500 pixels (which represents the user is only interested on viewing the top results) and returned back to its original place 6 seconds later (A quick view of less than 10 seconds is assumed experimentally to see the searching results). Next, one of the links in the home page is clicked. This task ends by scrolling down the newly loaded page for 500 pixels (which expresses that the user is trying to view the top of loaded page). For reasons of brevity, details of scrolling and delays are not mentioned in the description of task 2 to task 10.

Task 2 is implemented to publish random text-only posts. Post title and body have 10 and 1000 characters, respectively. Some users in WordPress just want to read other's posts. Task 3 is applied to browse through the WordPress contents. In addition to browsing, unsuccessful login is also covered in this task to simulate a user who forgets his username or password. Task 4 browses a specific post for which it leaves a comment. Replier name, email, website and the comment left, have 33, 18, 34 and 300 characters, respectively. In Task 5, a so called *core-task* is executed five times. Essentially, *core-task* sends a request to open the WordPress landing page and then closes the Firefox window after being successfully loaded. In Task 6, in addition to text, a single photo for each post is uploaded. Post title and text have 33 and 700 characters, respectively with a 640*480-pixel photo whose size is about 100KB. Moreover, each post is published on five different categories.

Task 7 is designed to delete posts that they previously published. By default, posts are sorted by publishing date and shown in multiple windows with 20 posts each. In this task, the first 20 published posts (which are actually the latest ones) are deleted. Further capabilities of this task are deleting the posts with specific publication date, changing the number of posts to be deleted, and deleting posts with 'draft' status. Another common task for WordPress users is to create a new post without sending immediately for further editing.

Task 8 is implemented to create a new post whose title and text are 32 and 1800 characters, respectively. The post remains in the dashboard and saved (i.e. 'draft' status) into 5 categories without being sent to the database. Task 9, adds a new user to the database with the following information: user-name (29 characters), e-mail address (59 characters), first name (9 characters), last name (10 characters), and password (9 characters). Moreover, possible roles, which are randomly assigned to each generated user include subscriber, administrator, editor, author, and contributer. Although task 4 has left comments for a post, those comments remain in the pending state until they get approved. Task 10 is designed to approve the 20 latest pending comments.

WPressClient generates three workload types:

(i) Read (called R), in which, executed tasks only read from database. For this type, workload is generated based on task 1, task 3 and task 5.

(ii) Write (named W), in which, database is updated. W-type workload is generated based on task 2, task 4, task 6, task 7, task 8, task 9 and task 10.

(iii) Read/Write (labeled R/W) in which all 10 tasks are executed.

For each type of workload, *WPressClient* selects tasks using round-robin technique from available task pool.

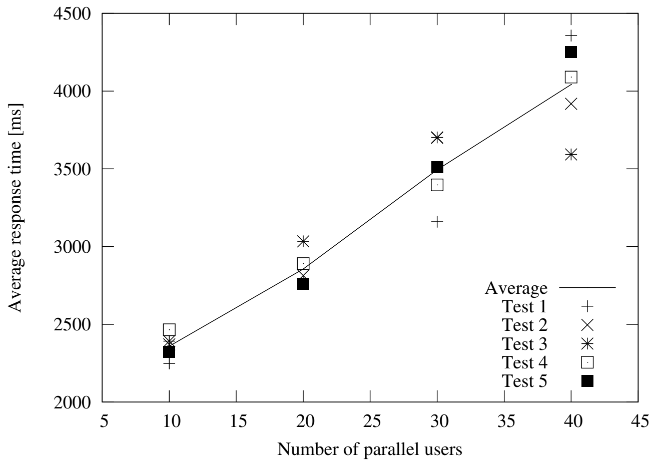
WPressClient is also responsible to measure the average response time and total operation cost of VMs on the clouds under test. In this chapter, *WPressClient* is executed on small and large instance sizes of Amazon EC2, Microsoft Azure and

Rackspace cloud. The small size instances have 1 CPU core, less than 1.75 GB of RAM, with $0.060 hourly cost. Nonetheless, the large size instances use 4 CPU cores, more than 7 GB of RAM, with more than $0.240 hourly cost (experimental environment will be discussed in more details in Section 3.1.3). *WPressClient* is configured to generate workload, based on available resources on the instance under test. The number of parallel users is obtained experimentally. Increasing number of parallel users leads to relatively large average response time particularly for *W* workloads in Amazon EC2. As the author is not testing a scalable framework for this benchmark, it is decided to go with 40 (for small size) and 150 (for large size) parallel users as the performance results up to these figures are sufficient to provide a good sense for the behavior of different cloud VMs.

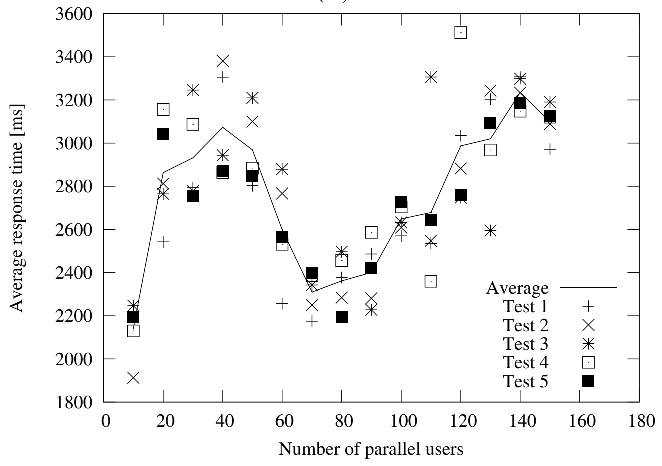
Each user is simulated by a single Java thread which runs a specific task (see Table 3.1). For example, the first 10 tasks for R-type workload would be: task 1, task 3, task 5, task 1, task 3, task 5, task 1, task 3, task 5, task 1 which are executed through 10 parallel Java threads. Task execution is repeated 5 times for any number of parallel users and the average response time is calculated. Moreover, total execution time of the whole experiment, i.e. completion time of 40 parallel users for small instances or completion time of 150 parallel users for large instances, is recorded by *WPressClient* for computing the total cost of VMs.

Just for illustration purpose, the five response times along with their average values of Microsoft Azure instances are shown in Figure 3.1. The values are adopted from the log files of Microsoft Azure with R-type workload. Figure 3.1(a) shows the difference between five response times in Azure small instance is linearly increased with number of users. However, significant fluctuations is observed between response times for Azure large instance (Figure 3.1(b)). Section 3.2 compares the response times in more details and discusses the main reasons for the observed fluctuations.

The systematic structure of *WPressClient* along with the distributed infrastructure discussed in Section 3.1.3 is an efficient and cost effective approach to make representative workloads for WordPress. Furthermore, it allows researchers and businesses to



(a)



(b)

Figure 3.1: Average response times for different number of parallel users in Microsoft Azure for Read Workload, (a) Small instances and (b) Large instances.

add new tasks or modify existing ones according to their requirements. *WPressClient* source code is available at [139].

3.1.3 Experimental Environment

The experimental environment used in this chapter is illustrated in Figure 3.2. It includes two parts: (1) server side that involves small and large instances on public cloud providers, and (2) client side which is a distributed infrastructure to run the *WPressClient* application.

On the server side, three well-known IaaS public cloud providers in the current market have been selected as representative samples for my benchmarking approach: (1) Amazon EC2, is the de facto standard and industry leader for public IaaS cloud

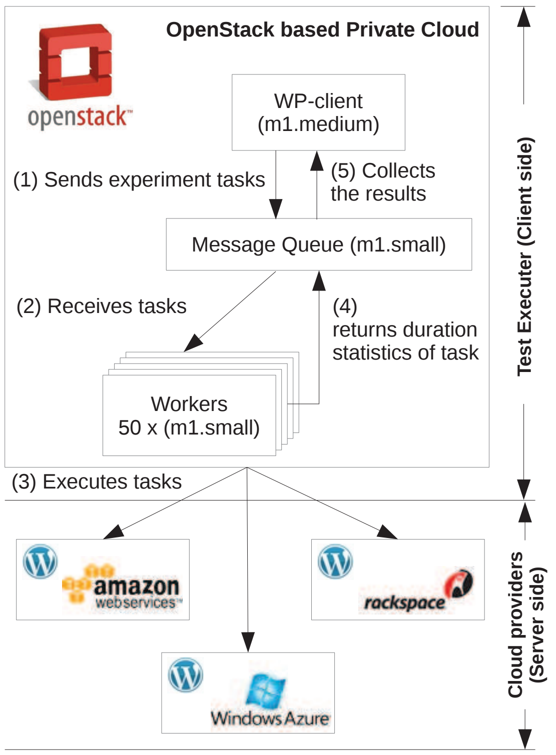


Figure 3.2: Experimental environment

providers. Its data-centers are distributed all over the world and various types of services are provided to handle different computing and storage demands. (2) Microsoft Azure also delivers a wide range of computing and storage services. It is making a huge effort toward being a prominent cloud provider by offering cost-effective services. (3) RackSpace Cloud is one of the biggest cloud providers in the United States of America (USA) with an easy to use control panel and high-quality customer service [27].

Two instance types of each provider are studied in this chapter (Table 3.2). For the sake of simplicity, they are shown by small and large labels in this chapter. Small implies m1.small in EC2, 1 GB standard in Rackspace, and (A1) Small in Azure. Large indicates m1.Xlarge in EC2, 8 GB standard in Rackspace, and (A3) Large in Azure. To achieve experiment fairness, corresponding instances are selected in a way to have comparable computing power, memory space, disk capacity and hourly cost. The Ubuntu server (version 12.04 LTS 64-bit), WordPress (version 3.5.2) and complete LAMP platform (Linux Ubuntu 64-bit, version 12.04; Apache HTTP Server, version 2.2.14; MySQL database, version 5.1.70 and PHP, version 2.3.2) are used on all six instances. The web, application, and database servers are located on the same

Table 3.2: Detailed Setup of Instances Deployed in the experiment

Name	Cores (ECUs)	RAM (GB)	Disk (GB)	Cost (USD/h)
Amazon EC2				
m1.small	1	1.7	160	0.060
m1.xLarge	4	15	1680	0.480
RackSpace Cloud				
1 GB Standard	1	1	40	0.060
8 GB Standard	4	8	320	0.480
Microsoft Azure				
(A1) Small	1	1.75	197	0.060
(A4) Large	4	7.0	412	0.240

Table 3.3: Detailed Setup of Instances Deployed in Test Executer

Instance Type	Processor Architecture	Virtual CPU (ECUs)	Memory (GB)	Disk (GB)
m1.small	AMD 64-bit	1	1	40
m1.medium	AMD 64-bit	2	3	40

instance in all tests. Therefore, WordPress uses the local database associated with the LAMP server.

The client side of benchmark is a distributed infrastructure, named *Test executer*, located in an OpenStack based private cloud. *Test executer* consists of 52 instances: (1) A m1.medium instance, *WP-Client*, which is the front-end used to trigger tests and collect the results. (2) A cluster of 50 m1.small instances, named *Workers*, is responsible to execute *WPressClient* application in the private cloud. (3) A m1.small instance, referred as *MQ*, runs an Apache ActiveMQ message queuing system.

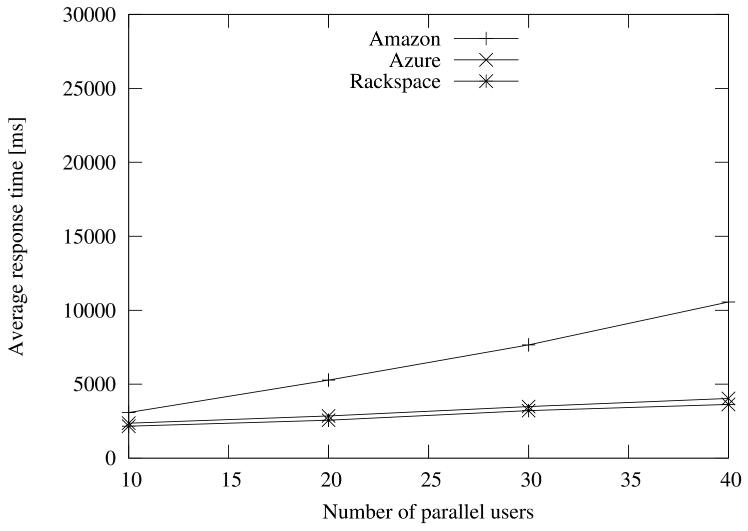
MQ enhances the extensibility of experiment by decoupling *WP-Clients* from the *Workers* so that *Workers* can be easily added or removed. To increase the number of *Workers*, it is only needed to start more instances which can hot-plug into the system running the experiment. Similarly, if one of the *Workers* crashes or turns off, it will neither get any further tasks nor break the experiment.

Table 3.3 depicts detailed setup of each instance used in *Test executer*. As illustrated in Figure 3.2, *Test executer* essentially involves five steps. First, *WP-Client* generates tasks and sends them to *MQ*. Each task is actually an execution of a single task in *WPressClient* application. Then, *Workers* receive tasks from *MQ* based on a first-come first-serve (FCFS) scheduling. After that, *Workers* run each task five times against each WordPress instance on Amazon EC2, Rackspace Cloud and Microsoft Azure. Next, each active *Worker* returns duration statistics of its running task to *MQ* and finally, *WP-Client* collects the results from *MQ*.

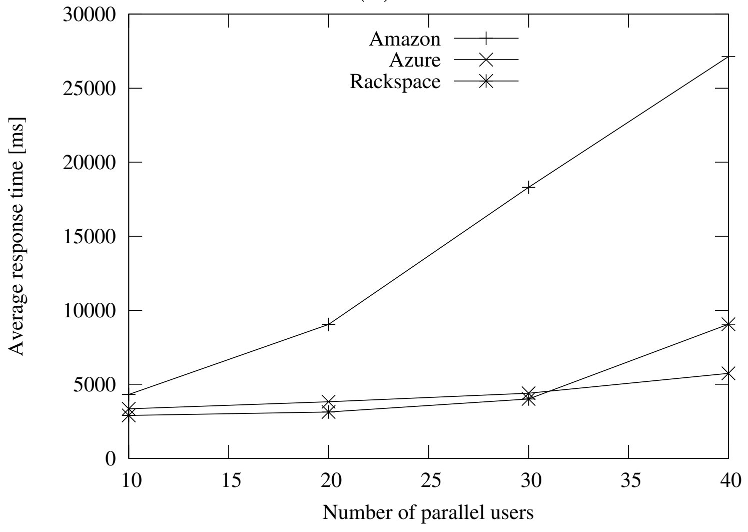
3.2 Results and Discussion

In this section, the results of applying *WPress* to Amazon EC2, Rackspace cloud, and Microsoft Azure are illustrated and discussed. In order to investigate the results further, the CPU micro-benchmark introduced in [74] is employed to demonstrate periods during which the VM is not allotted CPU time by the hypervisor. All the experimental results illustrated in this section were obtained from October 2013 to February 2014. Figure 3.3 shows that for small instances the response time is increasing with the number of parallel users regardless of the workload type. However, substantial fluctuations are observed for large instances (Figure 3.4). The author believes that undesirable effect of having no CPU affinity in *WPress* leads to inconsistency among large instances. Each VM in small instances has only one virtual core, hence the only scheduling overhead is at the hypervisor level to allocate physical CPU to the virtual core. However, for large instances, in which each VM possesses 4 virtual cores, an extra overhead is introduced in the virtualized layer for core scheduling.

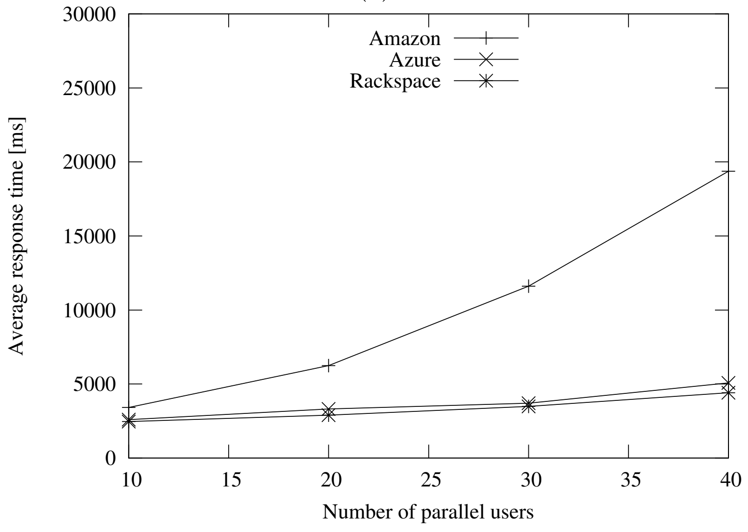
Figure 3.5 illustrates total VM cost for each cloud provider based on instance and workload types. The VM cost model is based on the total execution time of the *WPressClient* application times the VM hourly cost on each provider. The statistical t-test for small instance (Figure 3.5(a)) shows that Amazon EC2 has the largest cost compared with the other two cloud providers. However, no significant difference is found between the costs of Azure and Rackspace. As for large instances (Figure 3.5(b)), the t-test results show that Azure has the lowest cost among three providers. This is due to the lowest VM cost of large instances in Azure compared with that of Amazon EC2 and Rackspace at the time of running this experiment. In addition, the



(a)

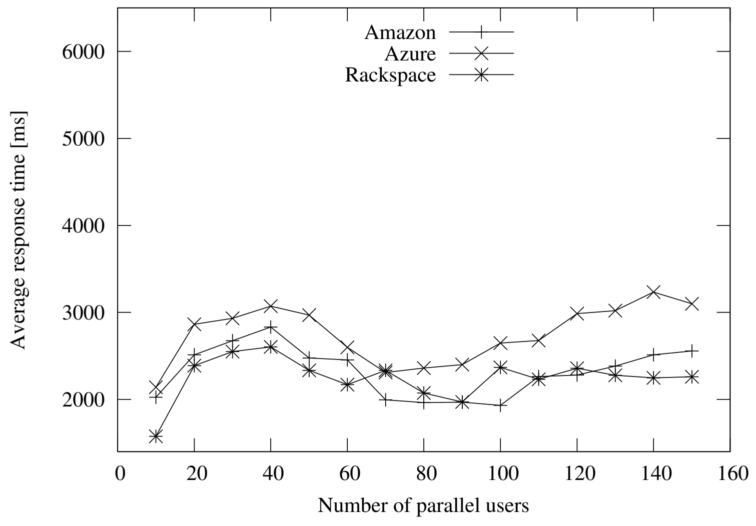


(b)

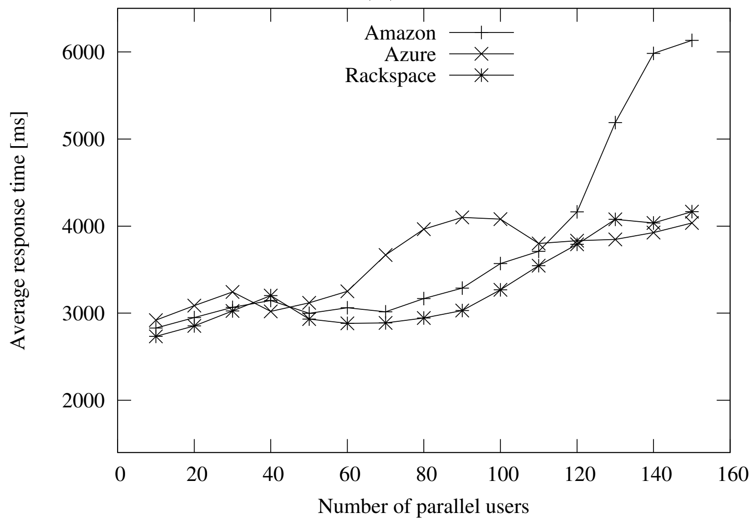


(c)

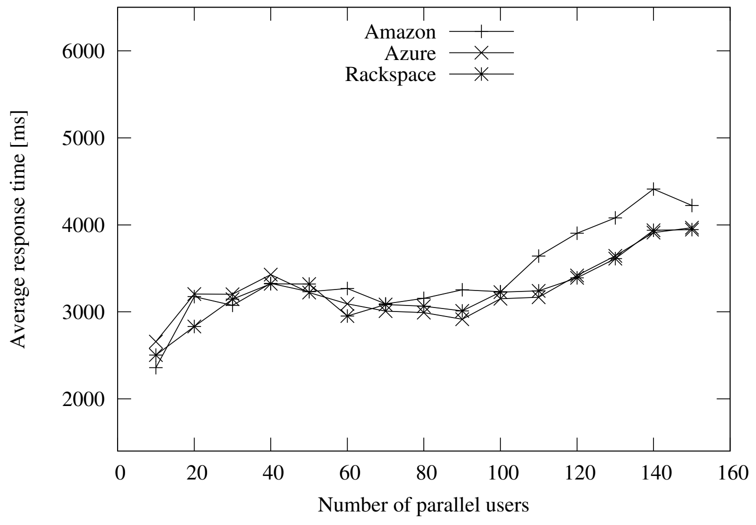
Figure 3.3: Average response times for small instances: (a) Read Workload, (b) Write Workload, (c) Read/Write Workload.



(a)



(b)



(c)

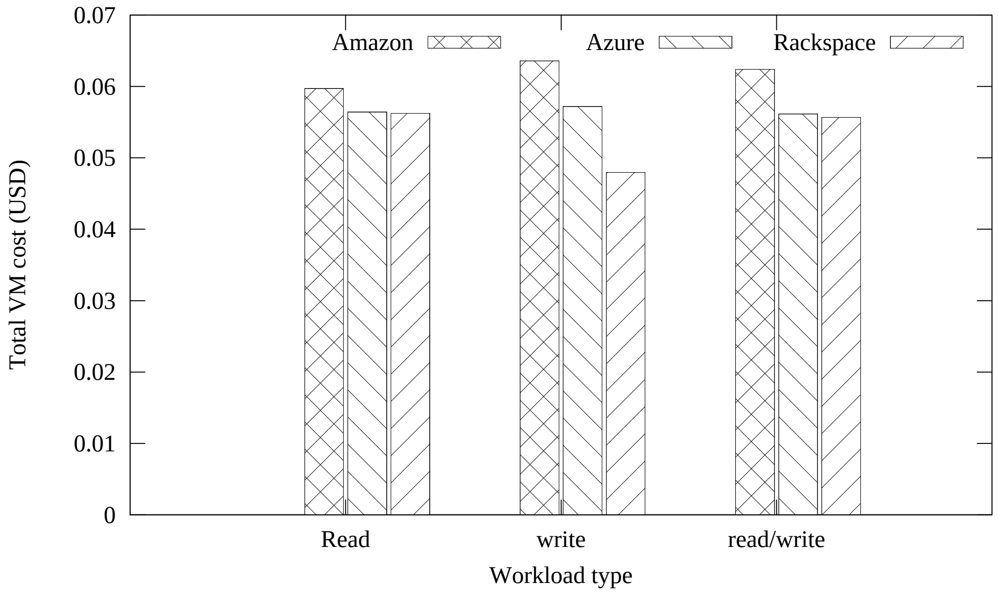
Figure 3.4: Average response times for large instances: (a) Read Workload, (b) Write Workload, (c) Read/Write Workload.

results show that Amazon EC2 and Rackspace are the most costly providers for large instances.

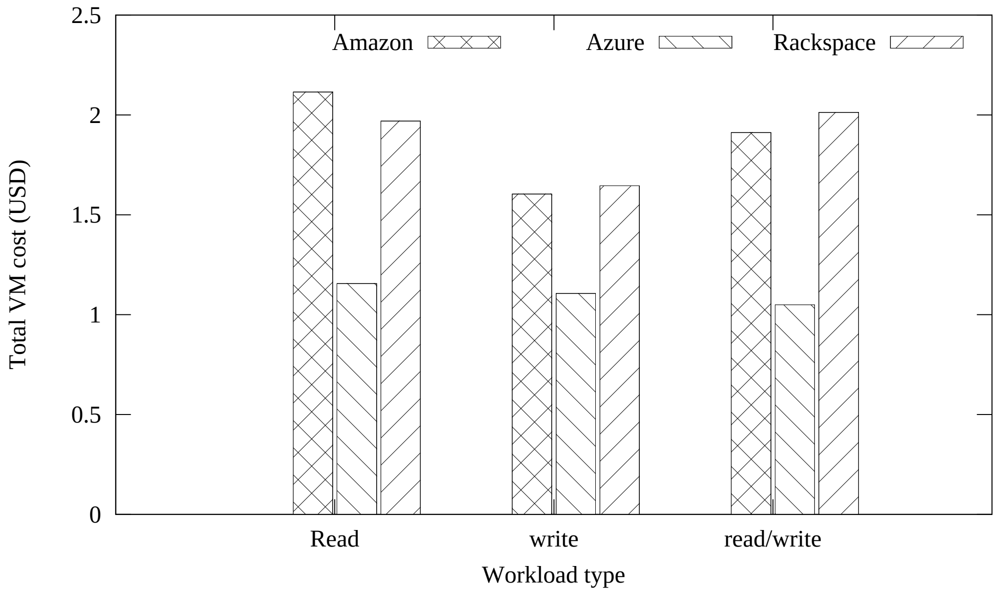
For further analysis, the CPU micro-benchmark proposed in [74] is employed to determine possible relationships between *WPress* results and CPU performance. The micro-benchmark runs on small and large instances of Amazon, Azure and Rackspace and continuously records CPU time for 1,000,000 times. The micro-benchmark application runs five times on each instance and labels *S1* to *S5* (*S* stands for *Sample*) are used to identify them. Moreover, the micro-benchmark is executed at different times (SGT) of a day (8am, 4pm and 11pm) to investigate any performance variations on the clouds under test. The micro-benchmark is the only running application on the instances. Therefore, it is expected to see the recorded values in a linear trend with positive slope, provided that allocated CPU to the instance is not shared with any other active instances.

Figures 3.6 and 3.7 depict sample outputs of small and large instances on Amazon, Azure and Rackspace recorded at 4pm (SGT) time-slot. The figure is zoomed in for the range of 200,000 to 400,000 loop numbers to magnify small changes for readers. The time stamps are measured in millisecond and are rather long (up to 13 digits). Therefore, for the sake of simplicity the values are normalized in the range of 1-10. In Figure 3.6(a), Amazon EC2 significantly shares allotted CPU with other instances. Vertical lines in Amazon results demonstrate period of time that CPU is not running the micro-benchmark application. On the contrary, Amazon large instances (Figure 3.7(a)) do not show significant CPU sharing. Similarly, considerable evidences of CPU sharing is not seen in small and large instances of Azure (Figures 3.6(b) and 3.7(b)), and Rackspace (Figures 3.6(c) and 3.7(c)).

The boxplot view of the observed micro-benchmark results are also depicted. Figures 3.8(a) and 3.9(a) show variation of 5,000,000 numbers recorded on each provider per time slot, whereas, Figures 3.8(b) and 3.9(b) illustrate variation of 15,000,000 recorded values for each provider. Due to noticeable CPU sharing, Amazon small instances have considerably larger time stamps than those of Azure and Rackspace

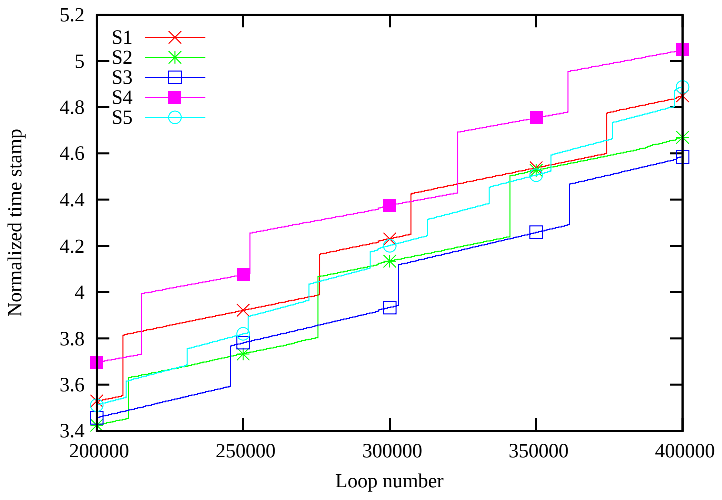


(a)

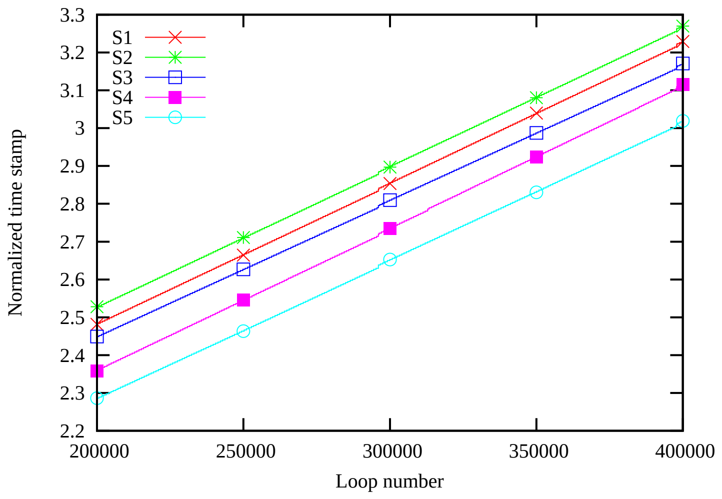


(b)

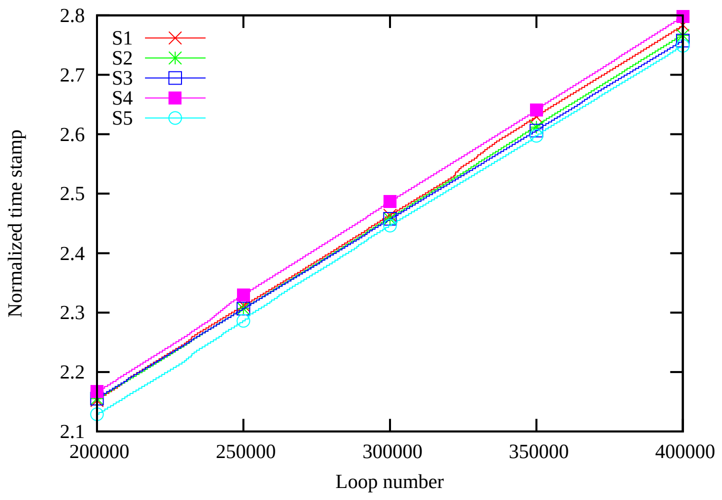
Figure 3.5: Comparing total VM cost for Read, Write and Read/Write workloads on different cloud providers. (a) Small instances, (b) Large instances.



(a)

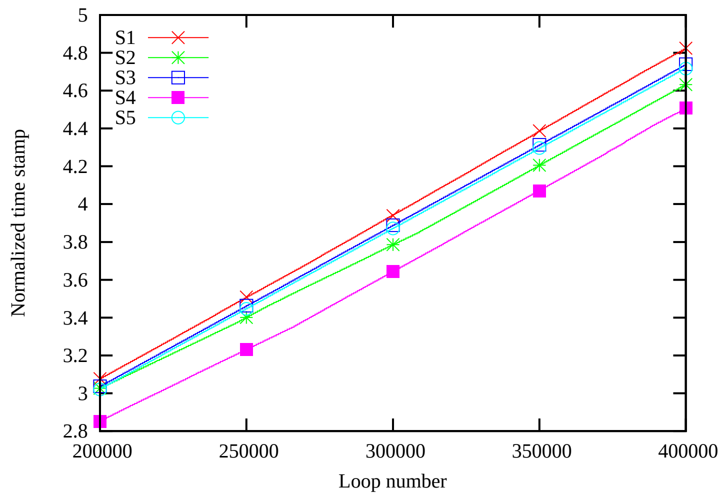


(b)

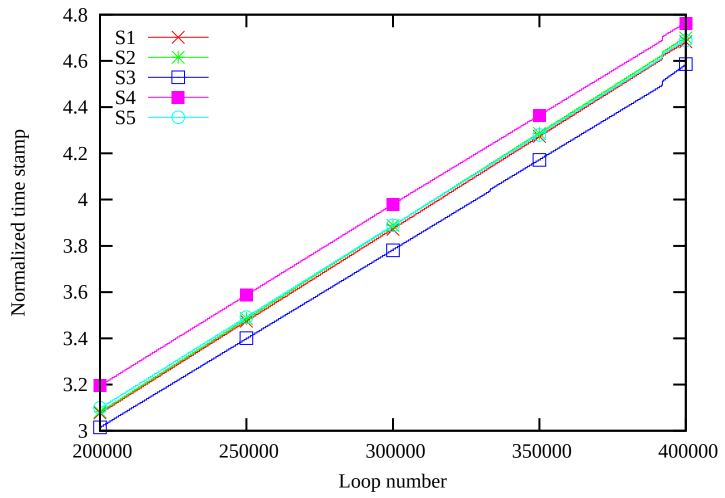


(c)

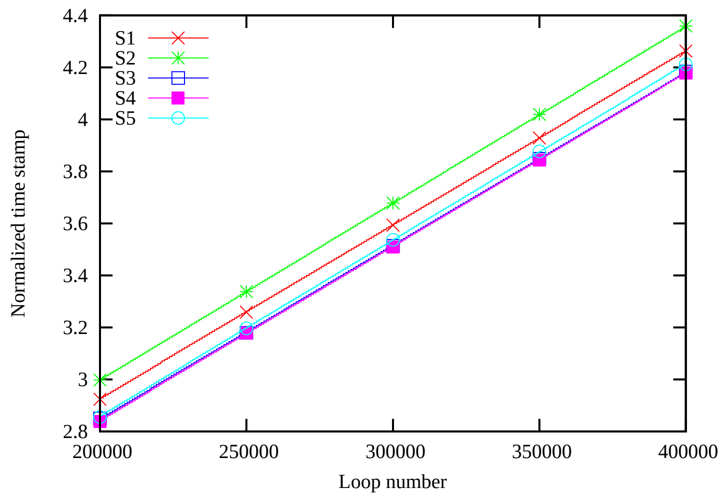
Figure 3.6: CPU micro-benchmark results observed at 4pm (SGT) for small instances: (a) Amazon EC2, (b) Microsoft Azure, (c) Rackspace Cloud.



(a)



(b)



(c)

Figure 3.7: CPU micro-benchmark results observed at 4pm (SGT) for large instances: (a) Amazon EC2, (b) Microsoft Azure, (c) Rackspace Cloud.

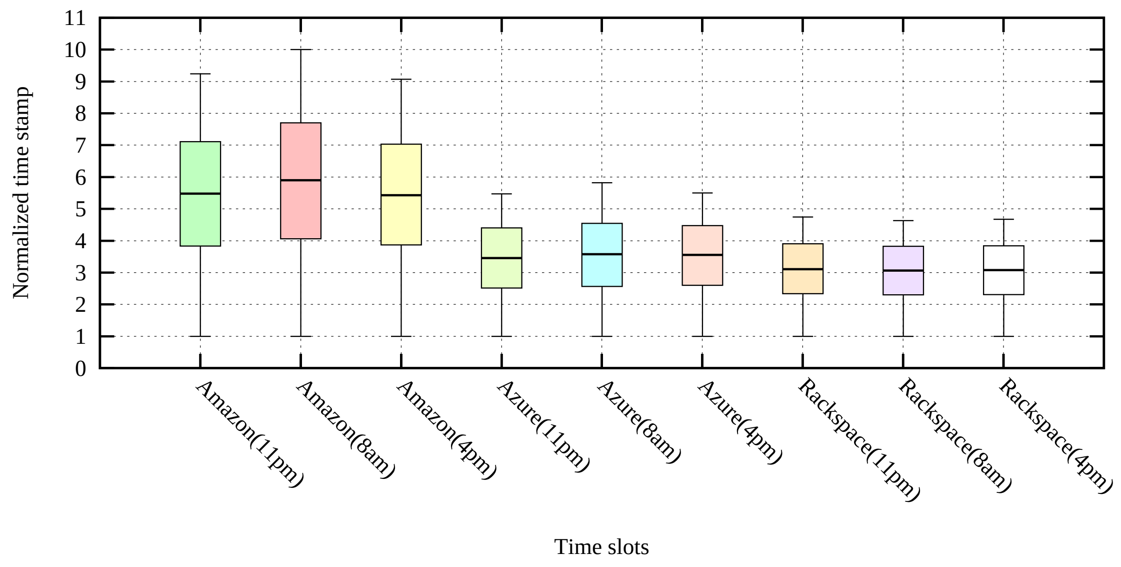
(Figures 3.8(a) and 3.8(b)). This accounts for drastically larger average response time of Amazon small instances.

Moreover, t-test is conducted to investigate variation among three samples on each provider (Figures 3.8(a) and 3.9(a)) as well as variation among different providers (Figures 3.8(b) and 3.9(b)). Results imply that all samples are significantly different with 95% confidence interval. In addition to the impact of CPU affinity, significant performance variations of CPU can be another reason for observed fluctuations of average response times in large instances. Obviously, CPU performance of large instances are better than those of small instances. It is worth mentioning, that the time stamps values shown in Figures 3.8, and 3.9, are normalized in the range of 1-10. The findings of this section are summarized as follow:

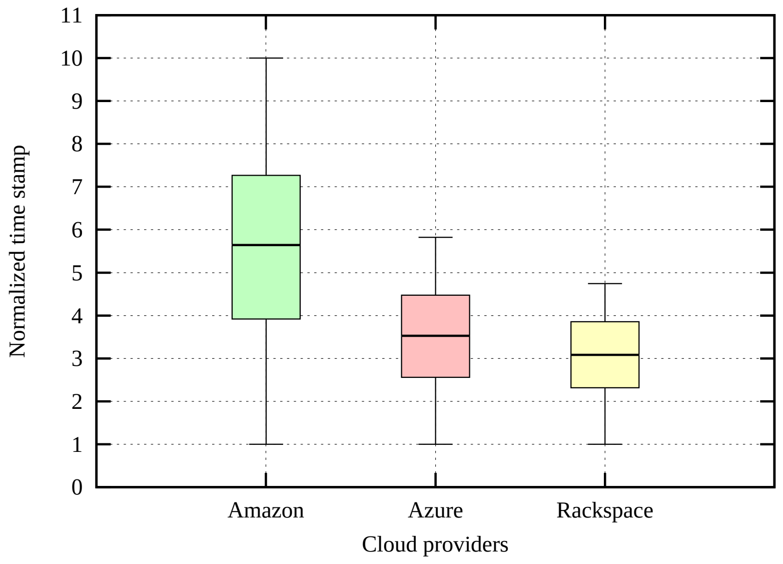
(i) For small instances, amazon EC2 has the largest average response times. This is attributed to the impact of CPU sharing, which is predominantly observed on Amazon small instances. Therefore, for a predictive performance, Amazon small instances should be used with care. However, it is observed that Rackspace has the lowest average response time and total VM cost compared with the other two providers. The high performance of Rackspace in small instances has been verified by CPU micro-benchmark results. Hence, Rackspace is the recommended Cloud solution for small instances.

(ii) Similarly, Rackspace shows the best performance for large instances among the three providers. Nonetheless, it is realized that the average response time between Rackspace and Azure is not statistically significant, especially for R/W workload. Thus, Azure is considered as a favorable alternative for the majority of customers according to its lower total VM cost.

Tables 3.4 depicts the recommended providers for small and large VM instances based on the experimental results obtained from October 2013 to February 2014.



(a)

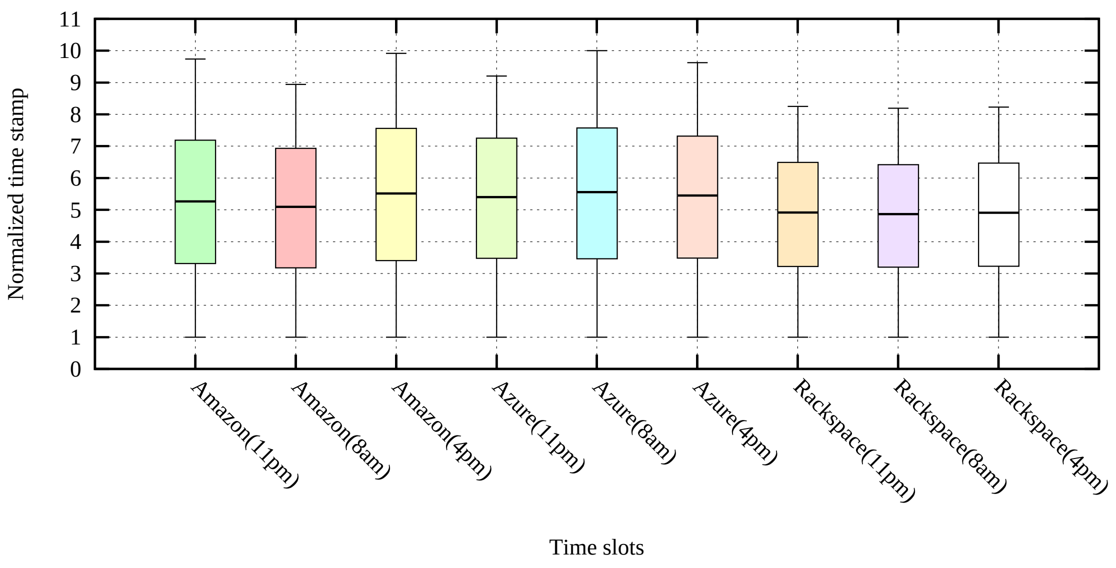


(b)

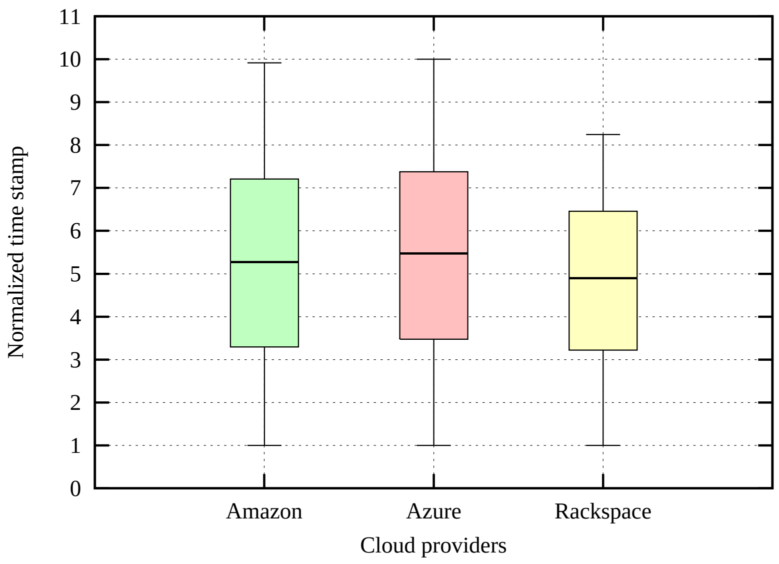
Figure 3.8: Performance variation of CPU on small instances: (a) for three providers on different time slots, (b) for three providers.

Table 3.4: Recommended Providers for Each Workload type on small and large instances.

Workload Type	Small VM Instance	Large VM Instance
Read (R)	Rackspace	Rackspace
Write (W)	Rackspace	Azure
Read/Write (R/W)	Rackspace	Azure



(a)



(b)

Figure 3.9: Performance variation of CPU on large instances: (a) for three providers on different time slots, (b) for three providers.

3.3 Summary

This chapter proposes *WPress* as an open-source performance benchmark for three-tier Web applications running on public clouds. *WPress* is tested for small and large VM instances of Amazon EC2, Microsoft Azure and Rackspace Cloud. Three different workload types (Read, Write and Read/Write) are considered. Furthermore, *WPressClient*, is implemented as an open-source Client and load generator application. WPressClient can generate representative workloads for WordPress and calculate the average response times and total cost of VMs for each provider. Moreover, the CPU micro-benchmark presented in [74] is employed to analyze CPU performance on the clouds under study and its impact on *WPress* results is identified.

The experimental results have shown that Rackspace Cloud and Microsoft Azure are the best Cloud solutions for small and large VM instances, respectively. Furthermore, it is noticed that average response times have substantial fluctuations on large instances that are due to undesirable effect of CPU affinity and significant performance variation of CPU. The benchmark results compared the stability of Amazon EC2, Microsoft Azure, and Rackspace Cloud in terms of handling concurrent requests to a three-tier web application running on them. However, it is not a very realistic assumption to have all tiers run on the same instance. In practice, each tier of a three-tier web application, such as WordPress, runs on a separate instance, and even on more than one instances for high availability purposes. Hence, a data center network needs to handle huge amount of network traffic between instances for an enterprise web application running on the cloud. Large amount of traffic has negative impact on the response time and energy consumption of the system.

The main focus of the next two chapters is to simulate more realistic scenarios for three-tier web applications running on the cloud and introduce a network-aware and power-aware VM migration heuristic in order to optimize the average response time and energy consumption of the system.

Chapter 4

Network-Aware VM Migration Heuristic for Improving the SLA Violation of Three-tier Web Applications on the Cloud

The powerful data centers with high bandwidth connections coupled with the need for on-demand computing and network resources on a pay-as-you-go basis make Cloud computing a suitable platform for three-tier web applications. However, the benchmark study conducted in Chapter 3 showed the importance of having an efficient resource optimization method for three-tier web applications running on the cloud. In fact, an efficient deployment of web applications on the Cloud environment needs to address two big challenges.

Firstly, the ever increasing number of servers and switches in modern data centers consumes considerable amount of energy leading to a rapid increase in total operational cost. For example, the energy consumption of IT infrastructures in the United States was about 61 billion kWh that corresponds to 2 percent of global carbon emission, and the numbers are likely to double every 5 years [70].

Secondly, the large number of web requests generated by millions of clients can easily overload the communication links inside the Data Center Network (DCN). The immediate consequence of traffic congestion on the DCN is an apparent increase in the average response time which may lead to SLAV. In particular, handling large number of requests in modern web applications, such as e-commerce, banking, and online shopping, involves complex database operations. This encompasses dynamic query processing and data generation which are remarkably slower compared to delivering static contents especially under heavy load [140].

The VM technology and VM migration are the accepted approaches to address the aforementioned challenges. To tackle the first challenge, the Modified Best Fit Decreasing (MBFD) VM placement algorithm proposed in [92] is exploited. The allocated hosts by the MDBF algorithm are experiencing the least increase in energy consumption after VM allocation.

Furthermore, the second challenge is addressed by suggesting an adaptive network-aware VM migration algorithm. The additional network traffic generated during VM migration may noticeably degrade the overall performance in network intensive applications. In order to mitigate this problem, the possibility of more than one VM migration on a congested link is not allowed. Moreover, the VM selection policy (VSP) considers the steady state amount of inbound/outbound traffic of VMs in order to reduce the undesirable effects of the migration overhead. More precisely, the following contributions are made in this chapter:

- An adaptive network-aware VM migration heuristic is proposed to improve the SLAV of three-tier web applications deployed in the Cloud. Once a congested link is detected, the VSP sorts all contributing VMs on the corresponding host. The sort is carried out in descending order in terms of the remaining steady state traffic of VMs. Afterwards, the Host Selection Policy (HSP), creates the list of target hosts with which the VMs are communicating. Next, HSP identifies the best migration pattern in order to minimize the SLAV and provide savings in energy consumption.

- Two host overloading detection polices are compared in HSP: 1) Static Threshold proposed by [92] which sets a static Upper Utilization Threshold (UUT) for all hosts, and 2) Adaptive Threshold in which the UUT is changed based on the average load of the system [141].

- CloudSim is extended to enhance its ability to simulate power-aware and network-aware VM allocation algorithms for web applications. The extended package, so called *power.network.datacenter*, integrates the *power* [121] and *network.datacenter* [109] packages of CloudSim. The package is also equipped with the three-tier model that is widely used in web applications.

The remainder of this chapter is organized as follows. Section 4.1 presents the system model used in the implementation of the heuristic. Section 4.2 describes the adaptive network-aware migration algorithm. Section 4.3 evaluates and analyses the obtained experimental results. Finally, Section 4.4 summarizes the results.

4.1 System Model

Without the loss of generality, a data center consisting of N heterogeneous physical servers (hosts) and one edge switch is considered. It is worth mentioning that the proposed algorithm in section 4.2 can work on more than one edge switches, but for the sake of simplicity all experiments are conducted with a single edge switch. Each host i is defined by the CPU performance measured in Million Instructions Per Second (MIPS), the amount of memory (RAM), and the total network capacity. The servers share a common Network Attached Storage (NAS) device in order to obviate the need to migrate the disk storage. The chapter is an attempt to consider a Platform as a Service (PaaS) Cloud environment that allows developers to deploy, host and run three-tier web applications regardless of their programming language and hosting frameworks.

The system consists of M heterogeneous VMs on data center resources. Each VM is characterized by its CPU requirement defined in MIPS, the amount of RAM and network bandwidth. The SLA is established between a service provider and a customer, to determine the costs and penalties based on the achieved performance level. SLAV can be caused by many events like VM consolidation, device (network or host) failure or even operator unavailability [142]. This chapter assumes a fault-free distributed system. The system with fault tolerant capabilities will be studied as future work.

Each host is equipped with a multi-core CPU which is defined as a CPU with n cores each having m MIPS with the effective total capacity of nm MIPS. This is rationalized as time-shared scheduling support allows applications and VMs to run on an arbitrary number of cores. It is assumed in this chapter that applications running on a VM cannot be arbitrarily parallelized. In order to achieve this goal, the CPU capacity requested by a VM is assumed to be less than or equal to the CPU capacity provided by each core.

4.1.1 Power Model

Energy consumption in data centers is mostly determined by servers, network devices, power supplies, and cooling systems [121], [1], [143]. In comparison to other system resources, it is shown that the main portion of energy is consumed by CPU [92]. The rapid increase in the usage of multi-core CPUs coupled with the complication of modeling the energy consumption of modern multi-core CPUs make developing a rigorous model a complicated research problem. Hence, instead of using an analytical approach, real data of energy consumption is utilized which is obtained as the results of the SPEC power benchmark. Two server configurations have been chosen: 1) HP ProLiant DL160 G5 (1 x [Intel Xeon L5420 2.5 GHz, 8 cores, 1 Gbps], 16GB) published in June 2008, and 2) IBM Server x3250 M3 (1 x [Intel Xeon X3470 2.93 GHz, 4 cores, 1 Gbps], 8GB) published in November 2009. The reason why these models have been chosen is that they are already provided as part of CloudSim and have enough resources for VM consolidation. The average active power values of the selected servers are presented in Table 4.1.

4.1.2 Network Model

The over-subscription of network resources and bandwidth sharing on DCN can have a significant negative effect on the performance experienced by network-intensive applications that may cause SLAV. Therefore, developing a representative model for

Table 4.1: Average active power of the selected servers at different load levels in Watts (W)

CPU utilization	HP ProLiant DL160 G5	IBM System x3250 M3
0%	148	41.6
10%	159	46.7
20%	167	52.3
30%	175	57.9
40%	184	65.4
50%	194	73
60%	204	80.7
70%	213	89.5
80%	220	99.6
90%	227	105
100%	233	113

DCN has a great deal of importance. Furthermore, in order to have an accurate performance evaluation of the migration algorithms, an efficient communication method must be taken into account. The network model within CloudSim [109], considers a flow model to capture the point-to-point steady-state behaviour of network transfers. Nevertheless, the flow model fails to capture the actual network latency. That is because the whole flow needs to be received by the switch before being forwarded to its destination. The inefficiency of the flow model becomes much more severe when a VM migration is triggered. To solve this problem, CloudSim is extended with the packet switching communication method which is used as a network model in the experiments. This provides the ability to model the actual network traffic more accurately.

4.1.3 Three-tier Web Application Model

Currently, most web applications (e.g. online shopping, online banking, social networking, etc.) are generated as three-tier systems because of its flexibility and software reusability [144]. This chapter considers the 3-tier model, that consists of presentation layer (Web tier), business logic layer (App tier), and database layer (DB tier) (Figure 4.1).

The Web tier receives requests from users, forwards its dynamic contents to the App tier, gets the results from App tier and displays them on the user's machine.

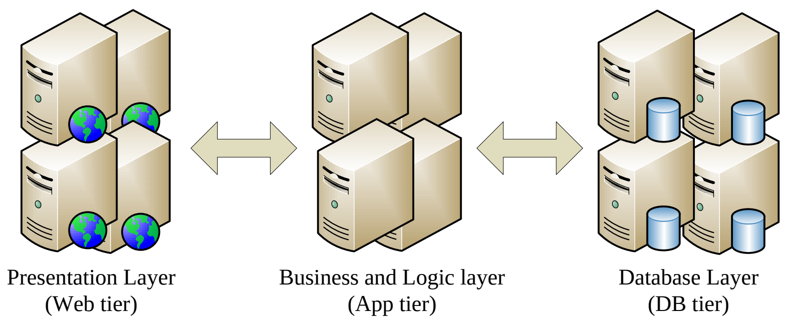


Figure 4.1: The 3-tier architecture

Apache and Microsoft Internet Information Server (IIS) are the widely used applications in the first tier. The App tier contains all the business logic for a web site. It processes the incoming requests from Web tier, renders the desired information obtained from DB tier, and sends the formatted results directly back to the Web tier. Apache Tomcat, Sun Java system application, and IBM WebSphore are well-known applications for the App tier. The DB tier is able to store several terabytes of text and multimedia data. Typical examples include Microsoft SQL Server, MySQL, and PosetgreSQL.

4.1.4 Workload Model

This dissertation employed the three-tier workload model proposed by [109]. The workload model presents each tier with a single Network Cloudlet (NC) and each NC consists of several stages. The Web tier NC is the starting point of a three-tier web application. It begins with sending stage (stage number = 0 in Web tier NC) representing sending some data to the Application tier NC. The application tier NC is in the network delay stage (stage number=0 in App tier NC) which expresses the state of waiting for some data to be received from Web tier NC. By sending data, the Web tier NC changes its stage from sending to network delay (Stage number = 1 in Web tier NC). This implies that Web tier NC is now waiting to receive a response from the Application tier NC. While data is being sent from Web tier NC to the Application tier NC, the Application tier NC is on receive stage (stage number = 1 in Application tier NC) denoting that it is receiving some data from the Web tier NC.

Once data is received successfully, the Application tier NC moves into stage change (stage number = 2 in Application Tier NC) and then enters the processing stage (stage number = 3 in Application tier NC). In the processing stage, the Application tier NC processes the received data from the Web tier NC and then gets into stage change (stage number = 4 in Application tier NC). Next, the Application tier NC enters the sending stage (stage number = 5 in Application tier NC) which accounts for sending some data to the Database tier NC. The database tier NC receives data from the Application tier NC (stage number = 1 in Database tier NC), processes the data (stage number = 3 in Database tier NC), and returns the results to the Application tier NC (stage number = 5 in Database tier NC).

While data is being processed in the Database tier NC, the Application tier NC is on the network delay stage (stage number = 6 in Application tier NC) meaning that it is waiting for some data from the Database tier NC. Once data is successfully received by the Application tier NC (stage number = 7 in application tier NC), an acknowledgment message is passed to the Database tier NC which is currently in the network delay sate (stage number = 6 in Database tier NC). By getting the acknowledgment message the Database tier NC enters the finish stage (stage number = 7 in the Database tier NC) and then the Database tier NC is removed from the NC list of Cloudsim. The application tier NC enters the stage change (stage number = 8 in the Application tier NC) and then to the processing stage (stage number = 9 in the application tier NC).

Once the data is processed by the application tier NC, the processed data is sent back to the Web tier NC (stage number = 11 in the Application tier NC and stage number = 2 in the Web tier NC). Once the data is received successfully in the Web tier NC, an acknowledgment message is passed to the Application tier NC so that the Application tier NC finishes its execution and subsequently it is removed from the list of NC in the CloudSim. The Web tier NC continues with processing the received data from the Application tier NC (stage number = 4 in the Web tier NC) and finally ends its execution (stage number = 6 in Web tier NC). Readers are referred to Garg et. al. [109] for more information about different stages in a NC.

The three-tier web requests are normally very small in size. Hence, huge number of individual web requests are needed to simulate a network intensive application which in turn demands for huge number of computing and memory resources. Because of our limitations in the computing resources, we used the concept of Aggregation Request (AR) in this thesis. The AR architecture (Figure 4.2) is nothing but the three-tier network model explained above. The only difference between AR and individual request is that we employed larger number of bits to be sent and received and longer processing time. In other words, each AR represents a collection of individual web requests. The concept of AR enabled me to use commodity computer systems to create huge amount of network traffic to evaluate my proposed network intensive migration algorithm for three-tier web application running on the cloud.

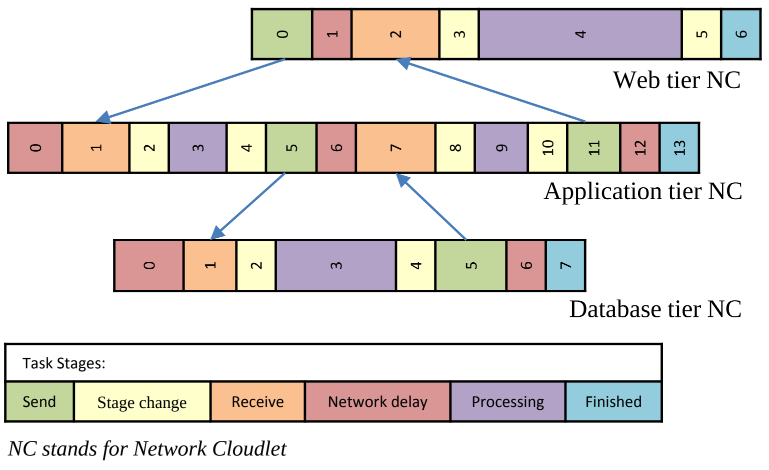


Figure 4.2: An Aggregated Request (AR) consisting of three Network Cloudlets (NCs)

4.2 Methodology

Recent advances in virtualization technologies have made it a common practice to dynamically consolidate VMs into smaller number of hosts. This enables the Cloud provider to switch idle nodes to the sleep mode in order to get rid of the idle energy consumption and reduce the overall energy consumption of the data center. In addition, an efficient VM migration plan can mitigate network congestion from the DCN which has a significant importance in network-intensive applications.

In this chapter, virtualization technology is used to efficiently mitigate the network congestion with the objective of minimizing the SLAV. The problem of minimizing SLAV by reducing the network congestion is very similar to the recently proposed α-Minimum Migration Cost Network Utilization (α-MCNU) problem [65]. In α-MCNU problem, the main objective is to maintain the utilization of all links below α such that the migration cost is minimized. The migration cost is defined as the total traffic generated due to migration. It is shown that α-MCNU problem is NP-Hard [65]. This section presents the heuristics to minimize the SLAV by reducing the network traffic.

The problem of VM allocation can be divided in two: the initial placement of VM on the host and the optimization of the current VM allocation. In this work, the recently proposed power-aware MBFD algorithm [92] is exploited to deal with the admission of new requests for VM provisioning and placing new VMs on hosts. Nevertheless, the optimization involves two steps: identifying the VMs that need to be migrated, and determining the destination host for migrating VMs. The pseudo-code of the algorithm is presented in Algorithm 4.1. The basic idea is to set UUTs for hosts and network links and prevent the total utilization of CPU and network from exceeding the specified thresholds. A hot switch is defined as a switch with at least one over-utilized link. The terms hot and over-utilized are used interchangeably in this thesis.

The network-aware heuristics VM migration algorithm is shown in Algorithm 4.1. The DCN traffic is continuously monitored on a real-time bases to determine the list of hot switches during each optimization cycle. The list is given to Algorithm 4.1 as an input. Essentially, the algorithm checks all *congested links*[1] of each hot switch (line 3). For each congested link, the algorithm finds all candidates for migration (line 8) and calculates their *Network Gain (NG)* (lines 12 and 18). The NG is calculated using equation 4.1, where the total traffic to be retransmitted (RT) after migration together with the resultant steady state traffic ($RSST$) is subtracted from the total steady state traffic that is removed from the DCN ($RFDCN$). The steady state traffic

[1] *Congested link* is a link whose bandwidth utilization exceeds 80% of the its maximum bandwidth capacity.

Algorithm 4.1 The network-aware VM migration algorithm

```
Input: $H_{switches} \leftarrow$ list of hot switches
Output: $M_{map} \leftarrow$ migration map
    initialization :
 1: $M_{map} \leftarrow \varnothing$
 2: for each hot switch $h_{switch} \in H_{switches}$ do
 3:     for each hot link $h_{link}$ of $h_{switch}$ do
 4:         $NG_{max} \leftarrow 0$
 5:         $host_d \leftarrow$ The host with direct connection to $h_{link}$
 6:         $L_{vm} \leftarrow$ list of VMs on $host_d$
 7:         for each VM $vm \in L_{vm}$ do
 8:             $L_{vm_c} \leftarrow$ list of VMs in communication with $vm$
 9:             $L_{host_c} \leftarrow$ list of hosts in communication with $vm$
10:             for each host $host_c \in L_{host_c}$ do
11:                 if $(NG_F(vm, host_c) > NG_{max})$ then
12:                     $NG_{max} \leftarrow NG_F(vm, host_c)$
13:                     $m_{candidate} \leftarrow \langle vm, host_c \rangle$
14:                 end if
15:             end for
16:             for each VM $vm_c \in L_{vm_c}$ do
17:                 if $(NG_F(vm_c, host_d) > NG_{max})$ then
18:                     $NG_{max} \leftarrow NG_F(vm_c, host_d)$
19:                     $m_{candidate} \leftarrow \langle vm_c, host_d \rangle$
20:                 end if
21:             end for
22:             if $(NG_{max} \neq 0)$ then
23:                 $M_{map} \leftarrow M_{map} + m_{candidate}$
24:             end if
25:             remove $h_{link}$ from $h_{switch}$
26:         end for
27:     end for
28: end for
29: return $M_{map}$
```

refers to the total amount of traffic that will be transferred between VMs. This is different from the instantaneous traffic between VMs which indicates to the amount of traffic at the migration time. The larger the NG, the higher the amount of traffic can be removed from the DCN and the lower SLAV is expected to be achieved.

$$NG = RFDCN - (RT + RSST) \tag{4.1}$$

The $RFDCN$ value is calculated by processing the traffic between the migrating VM and VMs on the destination host. In particular, the algorithm processes NCs that are running on the migrating VM. It also processes the NCs on the destination host that are communicating with the migrating VM.

For example, assume an application tier NC is being processed on the migrating VM with the current stage number of 6 (for more details about stage numbers please refer to section 4.1.4), and the VM migration is still in progress. Stage number 6, represents the network delay stage which means either that the data is still being sent from the application tier NC to database tier NC, or that the data is successfully sent by the application tier NC, but it is waiting to receive the response from database tier NC. In the former case, provided that the VM is migrating to the same host that runs database tier NC, the remaining data to be sent to DB tier NC is added to $RFDCN$. If the VM is migrating to the same host that web tier NC is being run, the traffic to be sent from application tier NC to web tier NC is considered as removed traffic. For the latter case, the remaining data to be received from database tier NC is added to $RFDCN$, if the VM is migrating to same host the runs DB tier NC. Similarly, the traffic to be sent from application tier NC to web tier NC is added to $RFDCN$, in case that the VM is migrating to the same host that runs the web tier NC. The same approach applies to all the other NCs.

The RT and $RSST$ values are calculated by processing the traffic between the migrating VM and VMs on other hosts. As stated for $RFDCN$ calculation, all NCs need to be processed. The same example is used to explain the calculation approaches. It is assumed that an application tier NC is being processed on the migrating VM

Table 4.2: Definition of variables in algorithm 4.1

Term	Description
M_{map}	The migration map
$H_{switches}$	Set of hot switches in DCN
h_{switch}	An individual hot switch
h_{link}	An individual over-utilized link (i.e. hot link)
vm_c	The candidate VM for migration
$m_{candidate}$	The candidate for migration
$host_d$	The host with direct connection to h_{link}
L_{vm}	Set of VMs allocated to $host_d$
L_{vm_c}	Set of communicating VMs with L_{vm}
L_{host_c}	Set of communicating hosts with L_{vm}
$host_c$	An individual member of L_{host_c}
NG_{max}	Maximum network gain
$NG_F(vm, host)$	Calculate the network gain for vm to $host$ migration

with the current stage number of 6 (i.e. network delay), and the VM migration is just completed.

Similarly, there are two possible cases: 1) if the sending of data from application tier to database tier is not completed, the amount of data that is received by database tier NC is considered as retransmitted traffic (RT) and the remaining data to be sent from application tier NC to database tier NC is considered as resultant steady state traffic ($RSST$). Moreover, the traffic to be received from database tier NC and the traffic to be sent from application tier NC to web tier NC are added to $RSST$, 2) if the data is successfully sent by the application tier NC to database tier NC, the amount of traffic that is received from database tier NC is added to RT and the remaining data to be received is added to $RSST$. Furthermore, the amount of data to be sent from application tier NC to web tier NC is added to $RSST$. The same approaches are used for all other NCs.

The algorithm identifies a $m_{candidate}$ for each congested link which is the best candidate for migration in terms of NG and eventually returns the migration map (M_{map}) for the current optimization cycle. Table 4.2 explains the variables used in algorithm 4.1.

Moreover, the system configuration after migration must satisfy the maximum capacities of all hosts such that the sum of the processing demands of all VMs allocated to a physical machine falls below the specified UUT. This chapter compares the efficiency of Algorithm 4.1 with two overloading detection policies: Migration with Static Threshold (MST) and Migration with Load-Balancing Threshold (MLB). The overloading detection policy is used to ensure that the destination host is not over-utilized after migration.

In MST policy, the UUT for each host is constant over time and is assumed to be 80% of the maximum CPU capacity of the host. However, the UUT in MLB policy is changing over time and is defined as the average CPU utilization of the system. On one hand, the MLB policy is expected to balance the CPU utilization of the system and consequently lower the average execution time of the requests; On the other hand, the MST policy is predicted to improve the SLAV due to its high flexibility in HSP. However, it is not very clear which policy will eventually lead to lower SLAV. To answer this question, an extensive simulation study is conducted in CloudSim and it is found that SLAV in MST is significantly lower compared to that of MLB policy. Please refer to Section 4.3.4 for more details.

4.3 Analysis

4.3.1 Performance Metrics

Several metrics are used to evaluate the performance of the proposed network-aware algorithm. The first metric is the percentage of SLAV, defined as the percentage of requests whose SLAs are violated. SLAV occurs when the response time perceived by the end user is larger than the specified value in the SLA contract. This can happen when a host is over-utilized and due to over-subscription of CPU. Consequently, a given VM cannot achieve its requested MIPS. Furthermore, network congestion can increase the response time for a request and lead to SLAV.

The percentage of SLAV represents the level at which the performance requirements negotiated between Cloud provider and user are not met. The Cloud provider is liable to pay a penalty for SLAV. The second metric is the average energy consumption of the physical hosts measured in kWh. The third metric is the average execution time of the simulation expressed in second (s). The average execution time and the percentage of SLAV are in close correlation. The lower the average execution time, the lower the percentage SLAV would be. The last metric is the average number of migrations generated by the algorithm during optimization of VM placement. Each VM migration may impose high network overhead to the data center. Hence, the author believes studying the average number of VM migrations can help to identify the efficiency of the proposed approaches.

4.3.2 Experiment Setup

The targeted system in this study is illustrated in Figure 4.3. In fact, conducting a reproducible experiment on a real data center to evaluate the suggested algorithm is incredibly difficult. Hence, simulation has been used to ensure the reproducibility of the experiments. CloudSim toolkit [23] is used as a well-known simulation framework for Cloud computing environment. CloudSim has been extended to enable combined power and network aware simulations as the main framework does not provide this feature. Apart from that, an extensible workload model for modern 3-tier web applications has been incorporated.

A data center with 24 heterogeneous physical hosts is simulated, half of which are HP ProLiant DL160 G5 servers, and the other half composed of IBM System x3250 M3 servers. The idea of choosing two power models is inspired by [121]. The reason why these models are chosen is that they are already provided as part of CloudSim. Moreover, it is experimentally determined that they have enough resources for VM consolidation. The configuration of the servers and their energy consumption are presented in Section 4.1.1.

Customers submit requests for provisioning of 60 heterogeneous VMs out of which 20 VMs are used as web servers, 18 VMs are chosen as application servers, and 22 are selected as database servers. The number of VMs allotted to the physical machine were determined experimentally. The VMs are initially allocated according to the VM instance type specifications assuming 100% CPU utilization. Nevertheless, during the lifetime, VMs consume less CPU resources according to the workload demand,

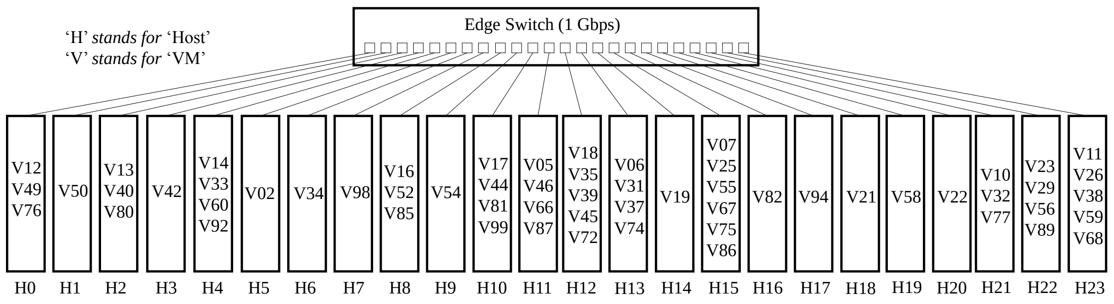


Figure 4.3: The simulated data center consisting of 24 hosts and 60 VMs connected by a 1-Gbps edge switch.

providing opportunities for dynamic VM consolidation. Four types of VMs are used, similar to what is used in [121] including [2500 MIPS, 0.85 GB], [2000 MIPS, 3.75 GB], [1500 MIPS, 1.7 GB], and [500 MIPS, 613 GB]. The configuration of VMs corresponds to the Amazon EC2 instant types [2]. The mere exception is that each VM has only one core, which is due to the fact that the workload trace used in this experiment is obtained from single-core VMs (Section 4.1.4). The specified threshold of the response time for measuring the SLAV is experimentally determined to be 50 seconds.

Without the loss of generality, it is assumed that DCN consists of a single 1 Gbps full-duplexed edge switch because implementing routing algorithms in CloudSim from scratch is a complex task that falls out of the scope of this research work. Each experiment was repeated 10 times and the average results are depicted in Section 4.3.4. As a future work, it is planned to evaluate the algorithm coupled with different routing algorithms on a typical 3-tier DCN architecture which is made up of edge, aggregation, and core switches.

4.3.3 Workload Data

In order to generate representative evaluation results, it is crucial to carry out the experiments based on server and traffic traces from real-world systems. It is necessary in CloudSim to have the CPU utilization for each NC. Nevertheless, workload traces of web applications [145,146] rarely contain such information. Hence, without the loss of generality, the PlanetLab workload trace files [121,147] are used in the experiment.

[2]Amazon EC2 Instance Types. http://aws.amazon.com/ec2/instance-types/

It is assumed that each workload trace represents the CPU utilization of one NC. Hence, three workload traces are used for a single AR. Furthermore, realistic traffic traces are used in the experiment. The system load is changed from 500 ARs to 5000 ARs. Moreover, it is assumed that each AR consists of 1800 individual requests.

Figure 4.4 illustrates the eight scenarios used in the experiment. The main rational behind the configuration of Figure 4.4 is to simulate a more realistic three-tier web application on which each tier runs on a separate VM (e.g. scenario 1) or even on multiple VMs (e.g. scenario 8). Eight scenarios are used to cover all possible combinations with respect to the number of VMs involved on each tier. For instance, the first scenario has one operational VM on each tier. This implies that all Web tier NCs are processed by VM 34, all App tier NCs are executed by VM 50, and all DB tier NCs are handled by VM 59. A more complicated, yet realistic, scenarios could consist of shared databases on which the App VMs communicating with more than one DB VM. An example of shared databases is implemented in scenario 8, in which four DB VMs (i.e. VM 23, Vm 29, VM 56, and VM 89) on host 22 (H22) are shared among four App VMs running on H13.

4.3.4 Simulation Results and Analysis

In this section, three algorithms are compared. The first one is called No Migration (NM), which does not apply any type of optimization methods. This implies the cases where VMs are not migrated even if the network is congested due to high demand. The second and the third algorithms are respectively MST and MLB, in which the network-aware optimization heuristic presented in Section 4.2 is used. All three algorithms benefit from the MBFD initial VM placement algorithm [92].

The simulation results of the average SLAV, average energy consumption, average execution time, and average number of migrations are presented. Figure 4.5 shows that an increase in the number of ARs has negative effect on the SLAV. This is mainly caused by the growing demand for network resources that can rapidly over-utilize the communication links and increase the response time. Moreover, an increase in the

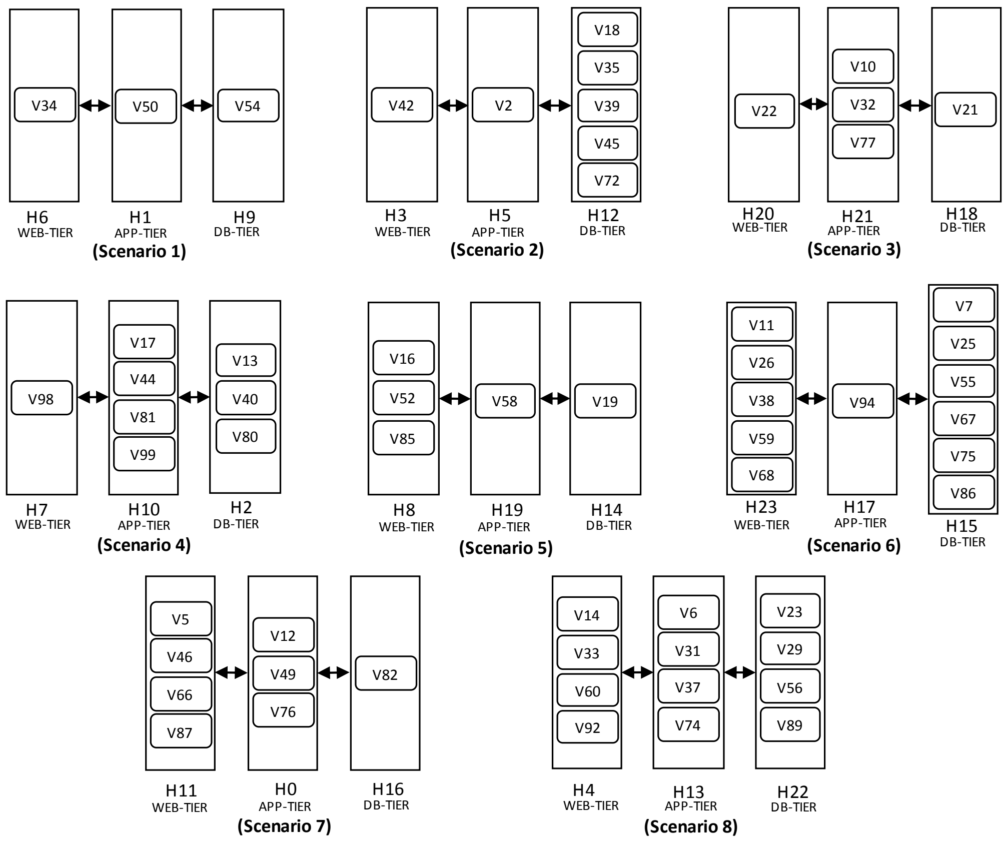


Figure 4.4: The simulated scenarios. In this Figure, "H" and "V" stand for Host and VM, respectively.

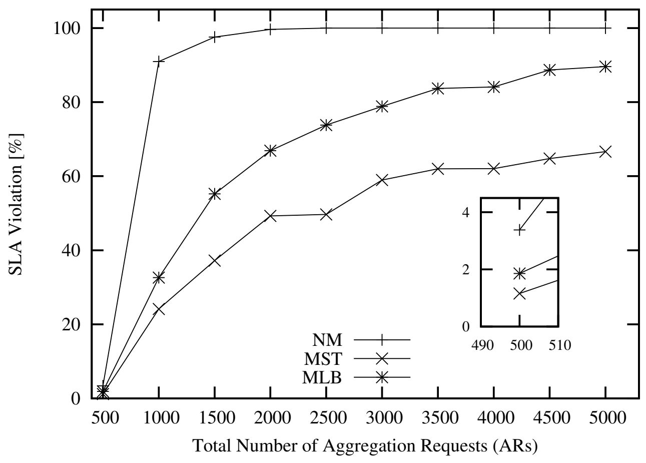


Figure 4.5: The comparison of the average SLA Violations. The figure is zoomed in to demonstrate the efficiency of the proposed network-aware algorithm, even for low load.

number of requests can lead to both communications and computation load, that subsequently results in longer completion time (Figure 4.6).

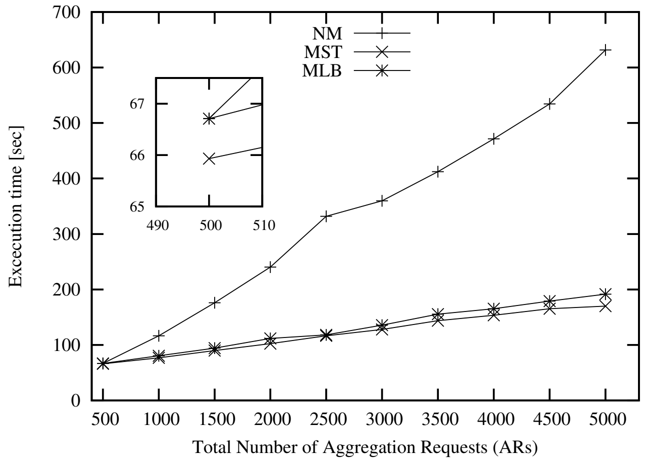


Figure 4.6: The comparison of the average execution time. The figure is zoomed in to demonstrate the efficiency of the proposed network-aware algorithm, even for low load.

Furthermore, the high load increases the energy consumed by the data center (Figure 4.7) because it increases the operational time of the servers. Additionally, increasing the load generates high network traffic. Hence, as shown in Figure 4.8, the optimization algorithm triggers more migrations in order to alleviate the network congestion. Figures 4.5, 4.6, and 4.7 are zoomed in to demonstrate the efficiency of the proposed network-aware algorithm, even for low load. The mean and 95% Confidence

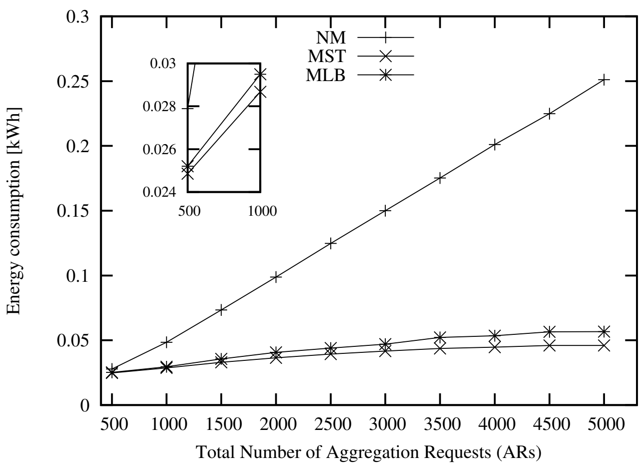


Figure 4.7: The comparison of the average energy consumption. The figure is zoomed in to demonstrate the efficiency of the proposed network-aware algorithm, even for low load.

Interval (CI) values of the SLAV, energy consumption, execution time, and number of migrations for the NM, MST, and MLB algorithms are presented in Table 4.3.

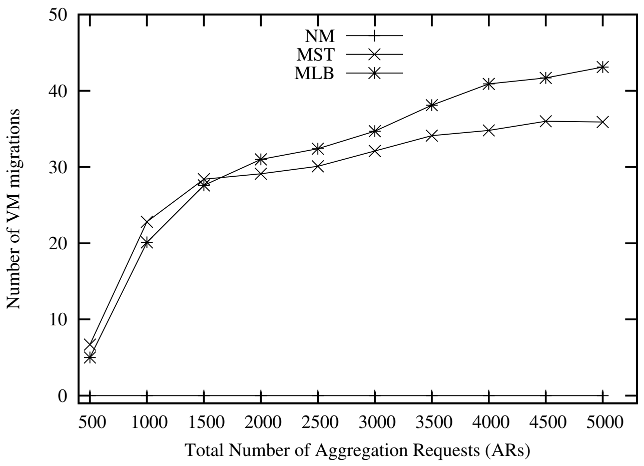


Figure 4.8: The comparison of the average number of VM migrations.

The author has conducted t-test to compare three algorithms and determine the one that minimizes the SLAV and energy consumption (Table 4.4). The results have shown that NM lead to a statistically significantly higher values of the SLAV, energy consumption, and execution time with the p-value < 0.0001. Furthermore, MST leads to significant reduction of SLAV and energy consumption compared to MLB. Nonetheless, there is no significant difference between MST and MLB in terms of execution time and number of migrations with the p-value $= 0.1238$ and p-value

Table 4.3: The final simulation results including the means and 95% CIs

Algorithm	SLA Violation	Energy Consumption	Execution Time	VM Migrations
NM	89.158	0.137	334.131	-
	(83.430, 94.886)	(0.123, 0.152)	(299.173, 369.088)	-
MST	47.590	0.038	121.187	29.00
	(43.433, 51.747)	(0.037, 0.040)	(113.909, 128.466)	(27.31, 30.69)
MLB	65.535	0.044	129.812	31.46
	(60.066, 71.004)	(0.042, 0.046)	(121.467, 138.157)	(29.22, 33.70)

Table 4.4: Comparison of different algorithms using paired *t*-test

Algorithms	Metric	Difference (means and 95% CIs)	*p*-value
NM vs MST	SLA Violation	41.569 (34.535, 48.602)	p-value < 0.0001
	Energy Consumption	0.099 (0.085, 0.113)	p-value < 0.0001
	Execution Time	212.943 (177.455, 248.431)	p-value < 0.0001
NM vs MLB	SLA Violation	23.623 (15.752, 31.494)	p-value < 0.0001
	Energy Consumption	0.093 (0.079, 0.108)	p-value < 0.0001
	Execution time	204.318 (168.599, 240.037)	p-value < 0.0001
MLB vs MST	SLA Violation	17.945 (11.118, 24.773)	p-value < 0.0001
	Energy Consumption	0.006 (0.003, 0.008)	p-value < 0.0001
	Execution time	8.625 (-2.380, 19.630)	p-value $= 0.1238$
	VM Migration	2.46 (-0.32, 5.24)	p-value $= 0.0829$

= 0.0829, respectively. Hence, it is concluded that MST leads to the minimum SLAV and energy consumption. This can be explained by the fact that some migrations are delayed by MLB algorithm such that the expected CPU utilization of the target host falls below the average utilization of the system.

The data obtained from log files of 5000 ARs is used to illustrate an example of the effect of MST and MLB on the migration time (Table 4.5). It can be seen from the table that MST is able to start migration of VM 82 (from Host 16 to Host 0) at

t = 26.48 s. This is the time at which the congestion is detected in the DCN and VM 82 is selected by VSP as a VM with the maximum migration gain. Nonetheless, the same VM migration can not start earlier than t = 55.31 s in the MLB algorithm. The reason for delay is attributed to the overloading detection policy used in MLB algorithm. The details are explained below.

Table 4.5: Comparison of MST and MLB in terms of migration time. VM migration starts at t =26.48 second in MST and t =55.31 second in MLB.

Description	Time (second)	MST	MLB
CPU utilization (Host 0)	t =26.48 s	42.81%	42.81%
	t =55.31 s	4.14%	11.98%
Upper Utilization Threshold	t =26.48 s	80.0%	5.87%
(UUT)	t =55.31 s	80.0%	14.0%

In Table 4.5, the CPU utilizations of Host 0 and UUT of the system are depicted at two times (t = 26.48 s and t = 55.31 s). The UUT in MST is 80% regardless of the time. However, it is 5.87% at t = 26.48 s and 14.0% at t = 55.31 s in MLB. Therefore, MST can start migration of VM 82 at t = 26.48 s because the host utilization is less than the UUT (i.e. 42.81% < 80%). However, CPU utilization in MLB exceeds the UUT of the system (i.e. 42.81% > 5.87%). Hence, MLB delays the migration to t = 55.31 s where the host utilization falls below the system threshold (i.e. 11.98% < 14.0%).

The overloading detection policy in MLB could result in the VM migration that not necessarily removes the maximum network traffic from the network. Hence, although the optimization algorithm still improves the SLAV compared to NM, the DCN remains congested for longer time compared to MST. This leads to more VM migrations in MLB than that of MST. Consequently, the overhead of more migrations in MLB increases the SLAV, energy consumption , and execution time compared to those of MST.

4.4 Summary

This chapter proposed an adaptive network-aware migration algorithm with the objective of minimizing the SLA violation for three-tier web applications. Moreover, the high energy consumption of data centers is addressed by using MBFD power-aware VM placement algorithm proposed in [92]. Furthermore, CloudSim is extended to enhance its ability to simulate energy and network aware VM allocation algorithms for three-tier web applications and is used to evaluate the algorithm.

Two host overloading detection policies are compared. First, the MST policy that sets a static upper utilization threshold for all hosts. Second, the MLB policy that considers the average load of the system as the upper utilization threshold. The experimental results and the statistical analysis clearly confirm that the network-aware migration algorithm coupled with either of MST or MLB can meet the main objective of this research work by providing significant improvement in terms of SLA violation and energy consumption.

The author wants to draw the reader's attention to the fact that MBFD algorithm is only applied for initial VM placement. The proposed network-aware heuristic identifies congested links, calculates the Network Gain (NG) for all candidate VMs, and migrates the one with the maximum NG, regardless of its effect on energy consumption. The next chapter extends the network-aware heuristic with energy-awareness capabilities in order to make further improvement in the energy saving of the system with negligible effect on the SLA.

Chapter 5

Combined Energy Efficient and Network Aware VM Migration Heuristics for Improving the SLA

Chapter 4, addressed the high energy consumption and network usage of three-tier web applications running on Cloud data centers. This chapter extends the network-aware migration heuristic presented in Chapter 4 with energy-awareness capabilities to make further improvements in the total energy consumption of the data center. In particular, once a congested link is identified, the algorithm monitors its traffic flows and calculates network gain and power gain values for all contributing VMs. Accordingly, the network gain list and power gain list are created for each congested link and eventually, the VM with the lowest sum of ranks is selected for migration.

In the case of multiple options with the lowest sum of the ranks two final selection policies are studied: 1) Network Priority (NP) policy and Power Priority (PP) policy. Moreover, two methods for calculating the power gain are compared: Power Balancer (PB) and Power Saver (PS). An extensive simulation study is conducted in CloudSim to achieve the best integration of final selection policy and power gain calculation method.

In summary, this chapter presents the following contributions:

- Firstly, the network-aware migration heuristic is extended to include energy-awareness capabilities. In particular, once a congested link is identified, the algorithm monitors its traffic flows and calculates network gain and power gain values for all contributing VMs. Accordingly, the network gain and power gain lists are created for each congested link and eventually, the VMs with the lowest sum of ranks, based on the network gain list and power gain list, are selected for migration.

- Secondly, to verify the effectiveness of the proposed heuristic, CloudSim version 3.0, a well-known cloud simulator, is extended with power-network awareness capabilities for three-tier web applications running on the Cloud.

The rest of the chapter is organized as follow. Section 5.1 presents the system model. In Section 5.2, the methodology is proposed and explained. Section 5.3 analyzes the results of experiments. Finally, Section 5.4 summarizes the findings.

5.1 System Model

A new power module is developed, which was integrated into the system model used in Chapter 4. In fact, three server types are used in order to highlight the efficiency of the proposed energy-aware technique.

5.1.1 Power Model

The physical host, network devices, and cooling systems are known as the key contributers of energy consumption in data centers [121], [1], [143]. Nonetheless, it is shown that physical hosts consume the larger portion of energy [92]. In fact, developing an accurate power model for modern multi-core CPU is a difficult task. Therefore, this chapter uses the actual energy consumption data obtained from SPECpower benchmark [148]. The proposed system consists of the following servers: HP ProLiant DL580 G5 (16 x [Intel Xeon processor L7345, 1860 MIPS], 16 GB, 1 Gbps), Dell Power Edge 2950 III (8 x [Intel Xeon E5440, 2833 MIPS], 8 GB, 1 Gbps), and Acer AC100 (4 x [Intel Xeon E3-1260L, 2400 MIPS], 4 GB, 1 Gbps). The average active power values of the selected servers are illustrated in Table 5.1.

It is worth mentioning that different servers and power models are used compared to those in Table 4.1. The main reason was to provide three different power models with significant variability of idle power consumption for each scenario in order to highlight the power-awareness efficiency of the suggested algorithm in this chapter.

Table 5.1: Average active power of the selected servers at different load levels in Watts (W)

CPU utilization	HP ProLiant DL580 G5	Dell Power Edge 2950 III	Acer AC100
0%	271	157	21.5
10%	280	173	24.3
20%	294	189	28.9
30%	309	204	31.9
40%	322	217	34.9
50%	335	230	38.9
60%	347	243	44.6
70%	359	253	47.5
80%	368	262	53
90%	376	270	54.6
100%	387	276	58

5.2 Methodology

New advances in virtualization technology has enabled consolidation of multiple VMs onto smaller number of physical hosts. Chapter 4, proposed an efficient VM management according to the DCN traffic in order to reduce communication delay of three-tier web applications. This chapter is an effort to enhance the proposed algorithm with energy-awareness capability in order to increase the energy saving in data center.

5.2.1 Energy and Network Aware VM Migration Algorithm

The basic idea of the proposed energy and network aware algorithm (see Algorithm 5.1) is to find the list of candidate migrations for each congested link, rank the list based on the NG and *Power Gain (PG)*, and select the candidate with the lowest sum of ranks for migration.

The operation of Algorithm 5.1 is illustrated in Figure 5.1. Essentially, the first step is very similar to the network-aware Algorithm proposed in Chapter 4. The only difference is that instead of finding the VM with the maximum migration gain, the network gain is calculated for all VMs and NG list (L_{NG}) is formed (Algorithm 5.1, lines 10-23). In step 2, the algorithm calculates the PG for the migration candidate corresponding to each entry of L_{NG} and forms the PG list (L_{PG}) (Algorithm 5.1, lines

Algorithm 5.1 The power and network aware VM migration algorithm

```
Input: $H_{switches} \leftarrow$ list of hot switches
Output: $M_{map} \leftarrow$ migration map
   $initialization$ :
 1: $M_{map} \leftarrow \varnothing$
 2: for each hot switch $h_{switch} \in H_{switches}$ do
 3:    for each hot link $h_{link}$ of $h_{switch}$ do
 4:       $L_{NG} \leftarrow 0$
 5:       $host_d \leftarrow$ The host with direct connection to $h_{link}$
 6:       $L_{vm} \leftarrow$ list of VMs on $host_d$
 7:       for each VM $vm \in L_{vm}$ do
 8:          $L_{vm_c} \leftarrow$ list of VMs in communication with $vm$
 9:          $L_{host_c} \leftarrow$ list of hosts in communication with $vm$
10:          for each host $host_c \in L_{host_c}$ do
11:             if $(NG_F(vm, host_c) > 0)$ then
12:                $m_{candidate} \leftarrow \langle vm, host_c \rangle$
13:                $NG \leftarrow NG_F(m_{candidate})$
14:                $L_{NG} \leftarrow L_{NG} + \langle NG, m_{candidate} \rangle$
15:             end if
16:          end for
17:          for each VM $vm_c \in L_{vm_c}$ do
18:             if $(NG_F(vm_c, host_d) > 0)$ then
19:                $m_{candidate} \leftarrow \langle vm_c, host_d \rangle$
20:                $NG \leftarrow NG_F(m_{candidate})$
21:                $L_{NG} \leftarrow L_{NG} + \langle NG, m_{candidate} \rangle$
22:             end if
23:          end for
24:          for each Migration $m_{candidate} \in L_{NG}$ do
25:             $PG \leftarrow PG_F(m_{candidate})$
26:             $L_{PG} \leftarrow L_{PG} + \langle PG, m_{candidate} \rangle$
27:          end for
28:          rank $L_{NG}$ and $L_{PG}$
29:          $m_{candidate} \leftarrow LSOTRF(L_{NG}, L_{PG})$
30:          $M_{map} \leftarrow M_{map} + m_{candidate}$
31:          remove $h_{link}$ from $h_{switch}$
32:       end for
33:    end for
34: end for
35: return $M_{map}$
```

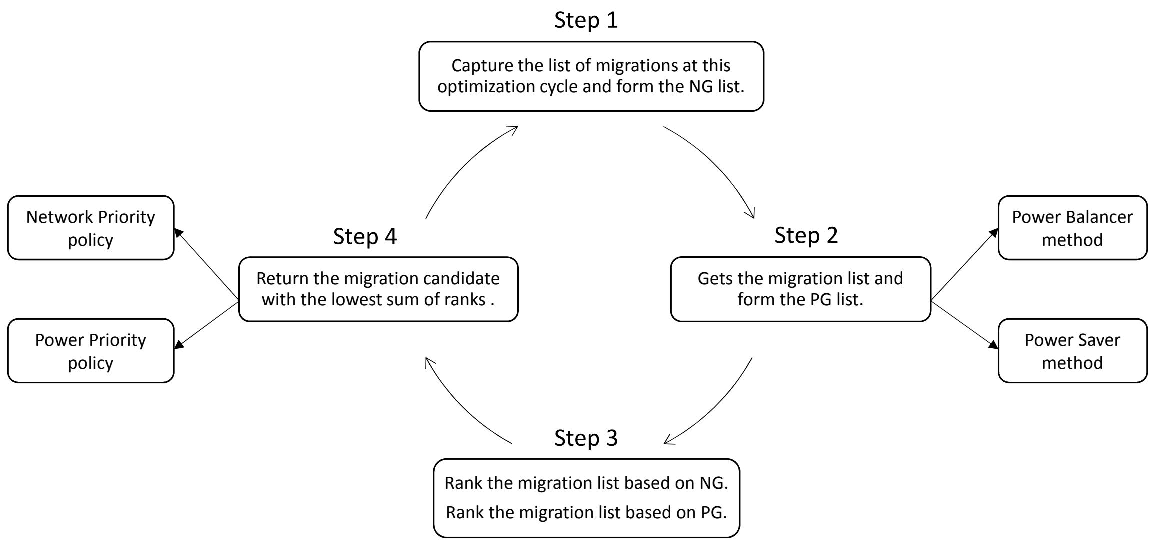

Step 1
Capture the list of migrations at this optimization cycle and form the NG list.
Step 2
Gets the migration list and form the PG list.
Power Balancer method
Power Saver method
Step 3
Rank the migration list based on NG.
Rank the migration list based on PG.
Step 4
Return the migration candidate with the lowest sum of ranks .
Network Priority policy
Power Priority policy

Figure 5.1: The operation of Algorithm 5.1. The PG in step 2 is calculated by two methods: Power Balancer method, and Power Saver method. Furthermore, if there are more than one candidate for migration in step 4, two policies are used to select the best candidate for migration.

Table 5.2: Definition of variables in algorithm 5.1

Term	Description
M_{map}	The migration map
$H_{switches}$	Set of hot switches in DCN
h_{switch}	An individual hot switch
h_{link}	An individual over-utilized link (i.e. hot link)
vm_c	The candidate VM for migration
$m_{candidate}$	The candidate for migration
$vm_{selected}$	The selected VM for migration
$host_d$	The host with direct connection to h_{link}
L_{vm}	Set of VMs allocated to $host_d$
L_{vm_c}	Set of communicating VMs with L_{vm}
L_{host_c}	Set of communicating hosts with L_{vm}
$host_c$	An individual member of L_{host_c}
NG_{max}	Maximum network gain
$NG_F(vm, host)$	Calculate the network gain for vm to $host$ migration
$PG_F(vm, host)$	Calculate the power gain for vm to $host$ migration
NG	The network gain
L_{NG}	The network gain list
PG	The power gain
L_{PG}	The power gain list
$LSOTRF$	return the candidate with the lowest sum of the ranks

24-26). In this chapter, the PG is determined by two different methods: the *Power Saver* method and the *Power Balancer* method (see Section 5.2.2). In step 3, the algorithm ranks L_{NG} and L_{PG} (Algorithm 5.1, line 28). In the last step, the algorithm returns the migration candidate with the lowest sum of the ranks (Algorithm 5.1, line 29).

In the case of having more than one migration candidates with the lowest sum of the ranks, two different selection policies are implemented to pick the best migration candidate: the *Network Priority* policy and the *Power Priority* policy (Section 5.2.3). Extensive simulation test has been conducted in Section 5.3.4.2 to identify the best integration of PG calculation method in step 2, and migration candidate selection policy in step 4. Table 5.2 describes the definitions of terms being used in Algorithm 5.1.

5.2.2 PG Calculation Methods

In this chapter, two different methods are implemented to calculate the PG. Figure 5.2 illustrates an example of changing in energy consumption before and after migration from the current host (source host) to destination host. It is assumed that a VM, vm, is migrating from the old host, $Host_A$, to the destination host, $Host_B$. The x axis indicates the simulation time in sec and the y axis displays the energy consumption in kWh. On the x axis, the t_1 and t_2 represent time before and after migration, respectively.

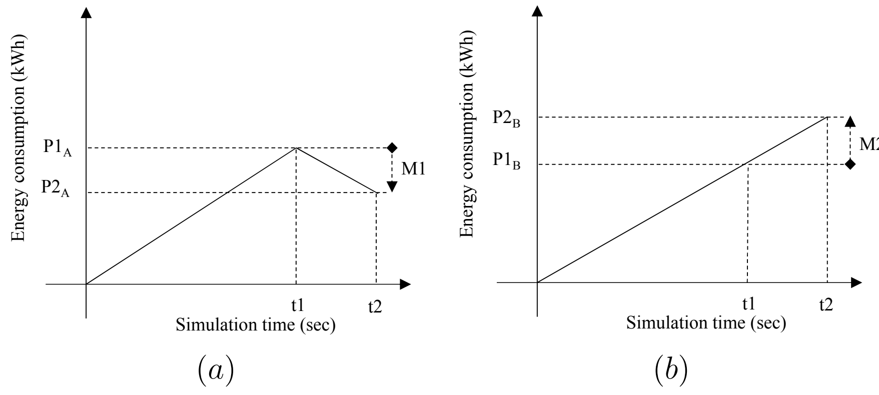


Figure 5.2: The change in energy consumption before and after migration. This example assumes migration of vm from $Host_A$ to $Host_B$. (a) Energy consumption on $Host_A$, (b) Energy consumption on $Host_B$

Once vm is removed from $Host_A$, its energy consumption drops from $P1_A$ at t_1 to $P2_A$ at t_2. However, resuming the vm on $Host_B$ increases its energy consumption from $P1_B$ to $P2_B$. It is worth nothing that the $M1$ and $M2$ are not necessarily equal because $Host_A$ and $Host_B$ are not homogeneous. The idle energy consumption of $Host_A$ is assumed to be A_{idle}.

5.2.2.1 Power Saver Method

Equation 5.1 shows a calculation of PG in the Power Saver (PS) method. The additional energy consumption imposed by vm on $Host_B$, i.e. $M2$, is subtracted from the saved energy due to removing vm from $HostA$, i.e. $M1$. Furthermore, A_{idle} is added to PG provided that vm is the last VM on $HostA$. This is because $Host_A$ can be turned off after migration. In this method, the larger the PG is, the greater energy saving can be achieved.

The calculation of $M1$ and $M2$ are based on the CPU utilization of the hosts and migrating VM. To get the $M1$ value, firstly, the CPU utilization on $Host_A$ namely U_A, is calculated. Next, the CPU utilization of the vm is captured right before stating the migration (U_{vm}). Using Table 5.1, U_A, and U_{vm}, the values of $P1_A$ and $P2_A$ can be calculated. Finally, the $M1$ is obtained as $P1_A - P2_A$. To get the value of $M2$, firstly, the CPU utilization on $Host_B$, called U_B, is calculated. Using Table 5.1, U_B, and U_{vm}, the values of $P1_B$ and $P2_B$ are obtained. Subsequently, the $M2$ is calculated as $P2_B - P1_B$.

$$PG = \begin{cases} M1 - M2 & \text{if } vm \text{ is not the last VM} \\ & \text{on } Host_A \\ (M1 - M2) + A_{idle} & \text{otherwise} \end{cases} \tag{5.1}$$

5.2.2.2 Power Balancer Method

The PG in Power Balancer (PB) method is determined by the actual energy consumption of hosts at t_2 (Equation 5.2). In particular, the actual energy consumption after migration on $Host_B$, i.e. $P2_B$, is subtracted from its corresponding value on $Host_A$, i.e. $P2_A$. Moreover, the A_{idle} is deducted from PG, in the case that vm is the last VM on $Host_A$ because the host can be turned off once the migration is completed. In the PB method, the smaller the PG is, the more energy is saved.

$$PG = \begin{cases} P2_B - P2_A & \text{if } vm \text{ is not the last VM} \\ & \text{on } Host_A \\ (P2_B - P2_A) - A_{idle} & \text{otherwise} \end{cases} \tag{5.2}$$

5.2.3 Migration Selection Policies

In fact, the final step in the proposed energy and network aware algorithm is to return the migration candidate with the lowest sum of ranks. However, it is possible to have more than one migration candidates with the lowest sum of ranks. An example is shown in Figure 5.3. The upper part of the figure depicts the ranked PG and NG lists. This example assumes that PG is calculated using the PS method. The lower part of the figure shows two migration candidates with the lowest sum of ranks: VM 115 from Host 18 to Host 16, and VM 65 from Host 16 to Host 18. Two selection policies are implemented for such examples to eventually obtain one migration candidate.

5.2.3.1 Network Priority Policy

As evident by its name, the Network Priority (NP) policy selects the migration candidate that is ranked ahead of others in the ranked NG list. In the example shown in Figure 5.3, the NP policy returns VM 115 from Host 18 to Host 16.

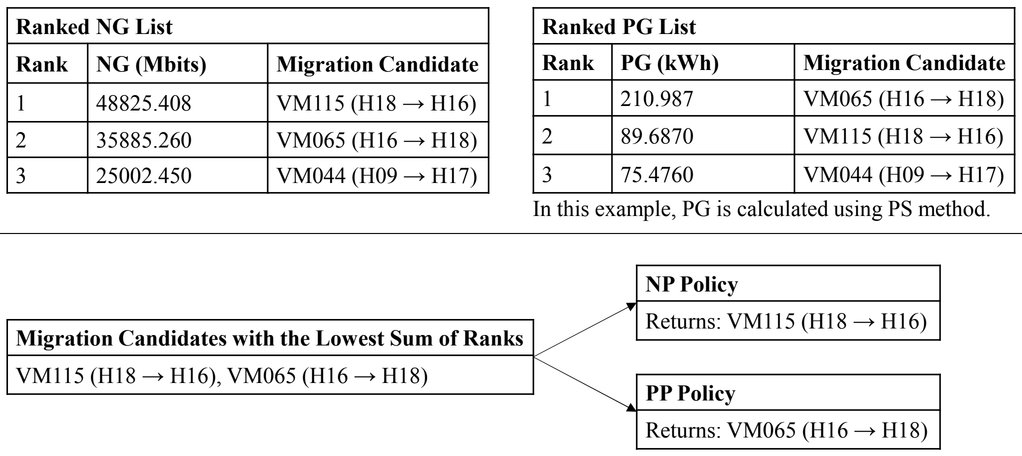
5.2.3.2 Power Priority Policy

In the Power Priority (PP) policy, the main preference is given to the PG. This policy returns the migration candidate that is ranked first among others in the ranked PG list. Hence, the output of step 4 in Figure 5.3 would be VM 65 from Host 16 to Host 18.

5.3 Analysis

5.3.1 Performance Metrics

This section defines the performance metric used in this study to evaluate the efficiency of proposed Algorithms. The performance metrics including average SLAV (in %),



Ranked NG List		
Rank	NG (Mbits)	Migration Candidate
1	48825.408	VM115 (H18 → H16)
2	35885.260	VM065 (H16 → H18)
3	25002.450	VM044 (H09 → H17)

Ranked PG List		
Rank	PG (kWh)	Migration Candidate
1	210.987	VM065 (H16 → H18)
2	89.6870	VM115 (H18 → H16)
3	75.4760	VM044 (H09 → H17)

In this example, PG is calculated using PS method.

Migration Candidates with the Lowest Sum of Ranks
VM115 (H18 → H16), VM065 (H16 → H18)
NP Policy
Returns: VM115 (H18 → H16)
PP Policy
Returns: VM065 (H16 → H18)

Figure 5.3: An example of having more than one migration candidates with the lowest sum of ranks

average energy consumption (in kWh), average completion time (in sec), average over-utilization time (in sec), and traffic gain (in Mbps) are described as follow.

- Average SLAV: represents the average percentage of requests whose average response time are longer than guaranteed values in SLA contract. SLAV may occur due to over-subscription of CPU on which VM's allocated CPU is lower than its requested amount. Furthermore, congestion in DCN increases the completion time of requests and consequently leads to SLAV. In fact, cloud providers are interested to reduce the SLAV as much as possible because they are obliged to pay penalty for SLAV.

- Average energy consumption (kWh): is used to evaluate the energy efficiency of algorithms. In this chapter the average energy consumption is a function of CPU utilization which is calculated based on the three linear power models presented in Section 5.1.1. The energy consumption of network devices will be studied as future work.

- Average completion time (sec): indicates the average time taken to complete the execution of ARs. Apparently, an AR is completed once its NCs have finished execution. The main reasons that increase the average completion time are the over-subscription of CPU and congestion in DCN.

- Average over-utilization time (sec): depicts the average duration of time during which, the average CPU utilization of physical hosts exceed 80% of their maximum computing capacity. The over-subscription of CPU is mostly observed on destination hosts once VM migration is completed. This is due to the fact that completion of migration increases NCs with almost no communication delay on destination hosts that consequently leads to high CPU utilization.

- Average traffic gain (Mbits): expresses the average amount of traffic that the optimization algorithm can remove from the DCN. In fact, the network-aware Algorithm suggested in Chapter 4 is expected to have larger average traffic gain compared to that of energy and network aware algorithm proposed in this chapter. The reason is that the former algorithm uses one optimization criterion and is always able to selects migration with the maximum traffic gain. However, the latter considers two criteria and may not necessarily choose the maximum traffic gain.

5.3.2 Experiment Setup

In practice, it is very difficult to conduct repeatable experiments in real data centers, which is particularly vital for comprehensive evaluation and comparison of the proposed algorithms [92]. Hence, simulation has been adopted in this chapter. CloudSim toolkit [23] is used to assess and compare the efficiency of the proposed heuristics. CloudSim has been extended with new packages to enhance it's energy and network awareness specifically for three-tier web applications.

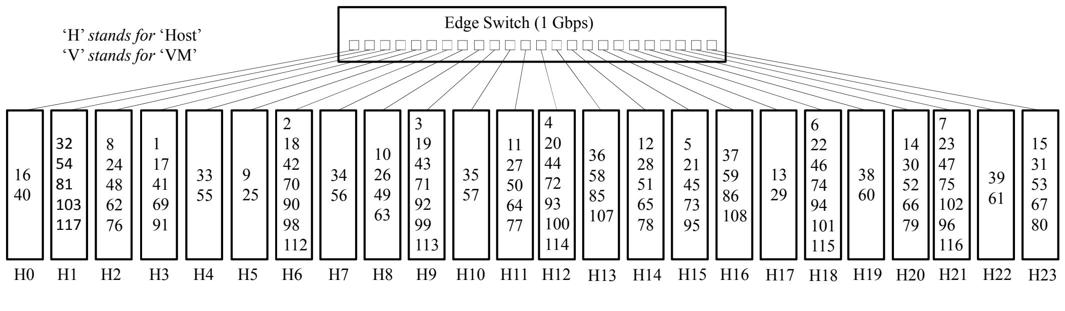


Figure 5.4: The simulated data center consisting of 24 physical hosts and 103 VMs interconnected by a Gigabit Ethernet edge switch.

Table 5.3: The host types in Figure 5.4

Server	HP ProLiant DL580 G5	Dell Power Edge 2950 III	Acer AC100
Host no.	H0	H1	H2
	H3	H4	H5
	H6	H7	H8
	H9	H10	H11
	H12	H13	H14
	H15	H16	H17
	H18	H19	H20
	H21	H22	H23

Figure 5.4 illustrates the architecture of data center in the simulation. This includes 24 heterogeneous physical machines which are adopted from three different server: HP ProLiant DL580 G5, Dell Power Edge 2950 III, and Acer AC100 (See Table 5.3). The servers are selected according to their computing capacity and their power consumption. In particular, it is aimed to have three levels of power consumptions: 271-387 Watt (high-level), 157-276 Watt (medium-level), and 21.5-58 Watt (low-level) to evaluate the proposed energy aware approach. The configuration of the selected servers along with their average active power consumption are presented in Section 5.1.1. The specified threshold of the response time for measuring the SLAV is experimentally determined to be 50 seconds.

The idea of choosing multiple power models is inspired from [121]. The reason why three models are chosen is that they are already available as part of CloudSim package. Furthermore, it is experimentally determined that they have enough resources for VM consolidation. The main reason for using different configuration compared to that of Figure 4.3 is to have three different power models with variable range of power consumption in order to signify the power-awareness efficiency of the proposed algorithm in this chapter.

This chapter simulates a three-tier web application consisting of 34 web server VMs (WEB-VMs), 34 application server VMs (APP-VMs) and 35 database VMs (DB-VMs). The number of VMs allocated to physical machines are obtained experimentally. The VMs are distributed across the physical machines, resulting in eight scenarios (Figure 5.5). The scenarios are distinguishable from one another by

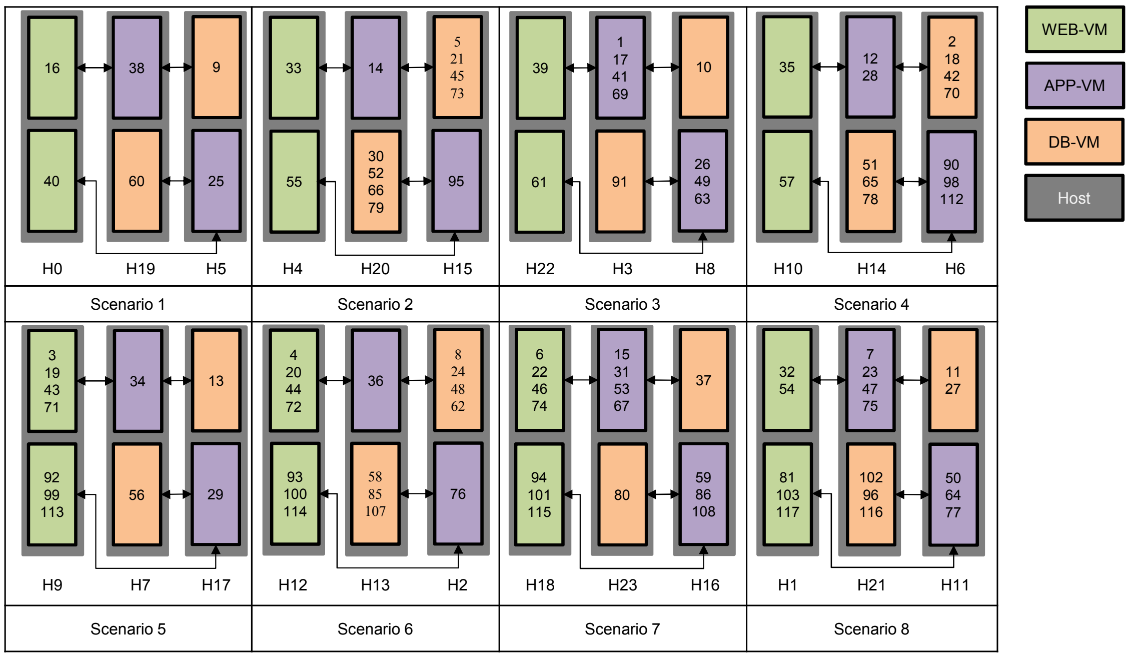


Figure 5.5: The eight scenarios simulated in this chapter. Each scenario contains at least two communication paths. For example, {VM 16 ↔ VM 38 ↔ VM 9} and {VM 40 ↔ VM 25 ↔ VM 60} are the two communication paths in the first scenario.

the number of VMs running on each tier. For example, in Scenario 1 there is only one VM for each tire which are VM 16 (WEB-VM), VM 38 (APP-VM), and VM 9 (DB-VM). A more realistic scenarios could consist of shared databases on which the APP-VMs are communicating with more than one DB-VM.

An example of such scenario, is Scenario 2 which has one WEB-VM (i.e. VM 33), one APP-VM (i.e. VM 14), and multiple DB-VMs (i.e. VM 5, VM 21, VM 45, VM 73). Different from Figure 4.4, each scenario is designed to have three physical hosts with different power models and at least two separate communication paths to increase migration opportunities with different PGs. For example, the communication paths in Scenario 1 include {VM 16 ↔ VM 38 ↔ VM 9} and {VM 40 ↔ VM 25 ↔ VM 60}.

Four types of VMs are used in the simulation which are already reported in the literature [121] with the exception that they have similar computing capacities: [1800 MIPS, 0.870 GB], [1800 MIPS, 3.840 GB], [1800 MIPS, 1.740 GB], and [1800 MIPS, 0.613 GB]. The focus of this chapter is to figure out the efficiency of proposed migration algorithms on reducing the SLAV and energy consumption. The main reason for fixed

MIPS is to cancel the effect of CPU on request completion time. The MIPS value is determined by an extensive set of experiments. The number of VMs allocated to physical machines are obtained experimentally.

As shown in Figure 5.4, the DCN is composed of a single edge switch. This is because designing a new routing algorithm in CloudSim and integrating that with the proposed migration algorithms is a complex mechanism which is out of the scope of this study. The edge switch interconnects a cluster of 24 hosts via full-duplex Gigabit Ethernet (GbE) links. As a future work, it is planned to evaluate the efficiency of the proposed migration algorithms in a more realistic network topology such as Fat-Tree made up of edge, aggregation, and core switches.

5.3.3 Workload Data

To achieve representative results, it is essential to conduct the experiment using real trace data. The available workload traces for web applications [145, 146] seldom contain the CPU utilization data that is required to be used for NCs. Therefore, PlanetLab trace files [121, 147] are used in the experiments. The system load is variable from 500 ARs to 5000 ARs and each experiment is repeated ten times. The average results are reported in Section 5.3.4.

5.3.4 Simulation Results and Analysis

This section compares the network-aware Algorithm proposed in Chapter 4 with the suggested energy-efficient network-aware algorithm in terms of average SLAV and energy consumption. The experimental setup in this chapter is different from that of Chapter 4. In order to have a fair comparison, the analysis consists of two parts. In the first part, the network-aware algorithm is compared against MST and MLB overloading detection policies. This confirms the experimental results shown in Chapter 4 that MST has better SLAV and energy consumption compared to that of MLB. Therefore, the second part adopts the MST overloading detection policy.

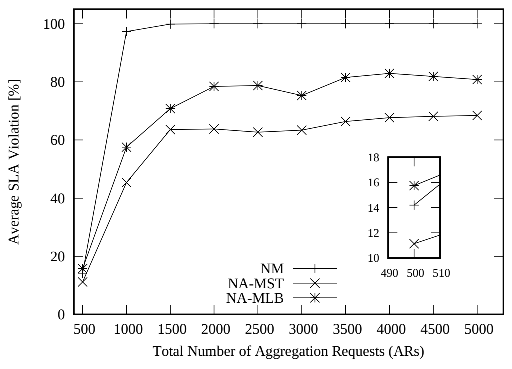
In the second part, the network-aware algorithm is compared with the energy and network aware algorithm. Moreover, the energy and network aware algorithm is evaluated to determine the most efficient power gain method and final selection policy. Similar to Chapter 4, all the experiments in this chapter use the MBFD algorithm [92] for the initial VM placement.

5.3.4.1 Network-Aware Algorithm

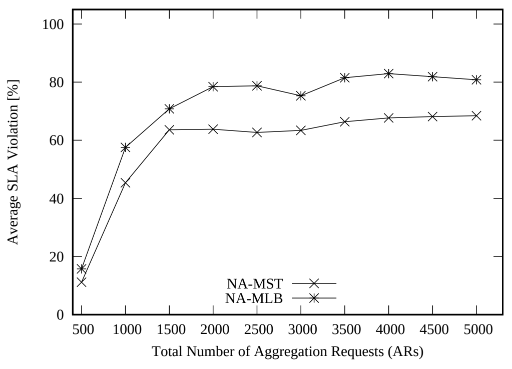
This part answers the question of which overloading detection policy performs best in terms of SLAV, energy consumption, and completion time. In particular three approaches are compared. The first approach, known as No Migration (NM), does not use any optimization method. The second approach (named NA-MST), uses the proposed network-aware optimization method with MST overloading detection policy. Finally, the third approach, called NA-MLB, uses the network-aware optimization method with the MLB overloading detection policy.

The simulation results, including the average SLAV, the average energy consumption, and the average completion time, are shown in Figure 5.6. The three plots in the left column, depicts the improvement of optimization approaches (NA-MST and NA-MLB) compared to NM and the the three plots in the right column compare the NA-MST and NA-MLB. An increase in the number of ARs negatively affects the average SLAV (Figures 5.6(a), 5.6(b)). The higher SLAV is attributed to rising demands for network resources and consequently higher network congestion. Furthermore, the increase in number of ARs enlarges the average energy consumption (Figures 5.6(c), 5.6(d)) and lengthens the average completion time (Figure 5.6(e), 5.6(f)). The mean and 95% Confidence Interval (CI) values of the average SLAV, average energy consumption, and average completion time of the three approaches are presented in Table 5.4.

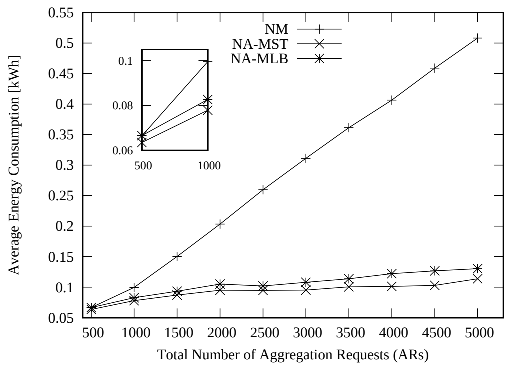
A statistical test (t-test) has been conducted to compare the three approaches. Table 5.5 shows that NA-MST ans NA-MLB approaches have significantly lower average SLAV, energy consumption, and completion time compared to NM approach with the p-value < 0.0001. Furthermore, it can be seen from the table that NA-MST shows significant reduction of average SLAV, energy consumption and completion



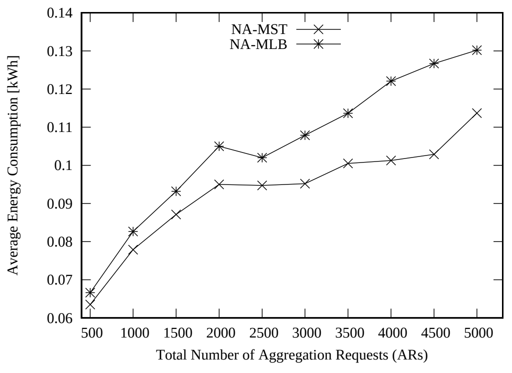
(a)



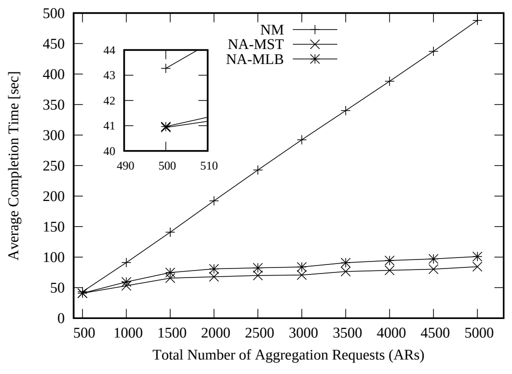
(b)



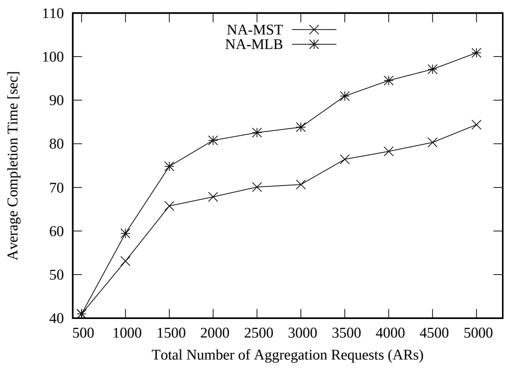
(c)



(d)



(e)



(f)

Figure 5.6: Comparing the efficiency of MST and MLB overloading detection policies in the proposed network-aware algorithm

Table 5.4: The simulation results including the means and 95% confidence intervals (CIs)

Algorithm	SLA Violation (%)	Energy Consumption (kWh)	Completion Time (sec)
NM	91.140	0.282	265.626
	(86.023, 96.257)	(0.254, 0.311)	(237.311, 293.942)
NA-MST	58.067	0.093	68.774
	(54.663, 61.472)	(0.090, 0.095)	(66.240, 71.308)
NA-MLB	70.379	0.105	80.588
	(66.449, 74.308)	(0.101, 0.108)	(77.062, 84.113)

time compared to that of NA-MLB. Hence, it is concluded that MST overloading detection policy is preferable to MLB. This is because, from the information contained in the log files, the UUT in MLB is much lower than that of MST. Therefore, MLB is not always able to select the VM with the maximum NG at each optimization cycle.

Table 5.5: Comparison of different algorithms using paired *t*-test

Group 1 = NM Group 2 = NA-MST		
Metric	Difference (means and 95% CIs)	*p*-value
SLA Violation (%)	33.072 (26.963, 39.180)	p-value < 0.0001
Energy Consumption (kWh)	0.189 (0.160, 0.218)	p-value < 0.0001
Completion Time (sec)	196.852 (168.598, 225.105)	p-value < 0.0001
Group 1 = NM Group 2 = NA-MLB		
Metric	Difference (means and 95% CIs)	*p*-value
SLA Violation (%)	20.760 (14.348, 27.173)	p-value < 0.0001
Energy Consumption (kWh)	0.177 (0.148, 0.206)	p-value < 0.0001
Completion Time (sec)	185.038 (156.680, 213.397)	p-value < 0.0001
Group 1 = NA-MST Group 2 = NA-MLB		
Metric	Difference (means and 95% CIs)	*p*-value
SLA Violation (%)	12.311 (7.143, 17.478)	p-value < 0.0001
Energy Consumption (kWh)	0.012 (0.007, 0.016)	p-value < 0.0001
Completion Time (sec)	11.813 (7.498, 16.128)	p-value < 0.0001

Table 5.6 illustrates an example of migration time, from the fifth run of 5000 requests. In this example, VM 58 is going to migrated from Host 13 to Host 2 using MST and MLB. The CPU utilization of the destination host (i.e. Host 2) and the UUT of system in both MST and MLB are depicted. It can be seen that at $t = 35.27$ s the CPU utilization (36.01%) is lower than UUT (80%) in MST and therefore the

Table 5.6: Comparison of MST and MLB in terms of migration time. Migration of VM 58 from Host 13 to Host 2 starts at t =35.27 second in MST. However, the same VM is migrated at t =88.34 second in MLB. CPU shows the CPU utilization of Host 2 and UUT stands for Upper Utilization Threshold.

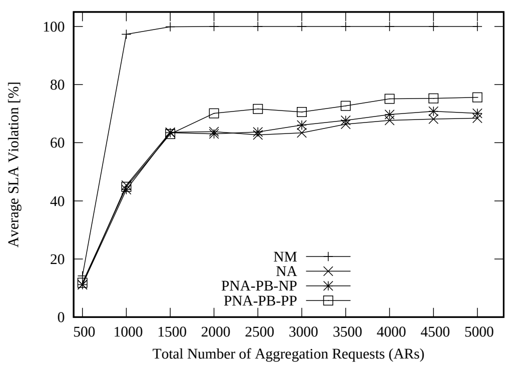
Time (sec)	MST	MLB
t =35.27	CPU = 36.01%	CPU = 1.42%
	UUT = 80%	UUT = 1.13%
	Migration is triggered	Migration is not possible
t =88.34	CPU = 1.09%	CPU = 2.01%
	UUT = 80%	UUT = 3.48%
	—	VM Migration is triggered

migration is triggered. This is the time at which migration of VM 58 from Host 13 to Host 2 has the highest NG. Nonetheless, at the same time MLB is not able to trigger migration of VM 58 because the CPU utilization of Host 2 (1.42%) is not lower than the UUT (1.13%). Instead, MLB triggers the VM migration at t = 88.34 s.

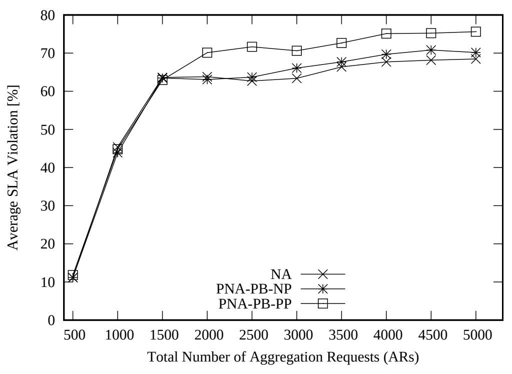
5.3.4.2 Energy and Network Aware Algorithm

This part presents the simulation results for the energy and network aware algorithm and aims to determine the optimized PG calculation method and final selection policy to be coupled in the algorithm. In the first stage, the PP and NP policies are compared for both PB and PS methods. The results show that NP is preferable in both methods due to its lower average SLAV, energy consumption and completion time. In the second stage, NP is adopted and PB is compared with PS. In the remaining of this chapter the MST overloading detection policy is adopted. This is due to the better efficiency of MST policy compared to MLB.

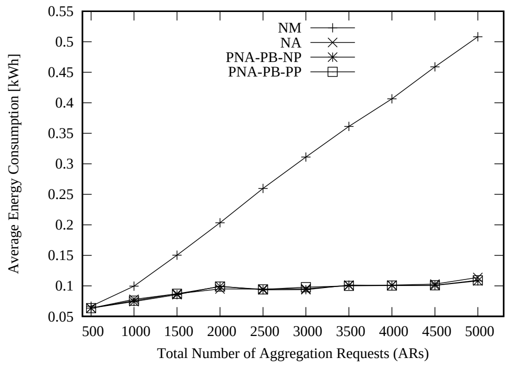
Three performance metrics are used in stage 1, namely, average SLAV, average energy consumption, and average completion time. This stage begins by studying the effect of final selection policies on the performance metrics whilst PB method is used. Four approaches are compared: the No Migration (NM) approach, the network-aware approach (NA), the power-network aware approach with PB method and NP policy (PNA-PB-NP), and the power-network aware approach with PB method and PP policy (PNA-PB-PP). The simulation results are depicted in Figure 5.7.



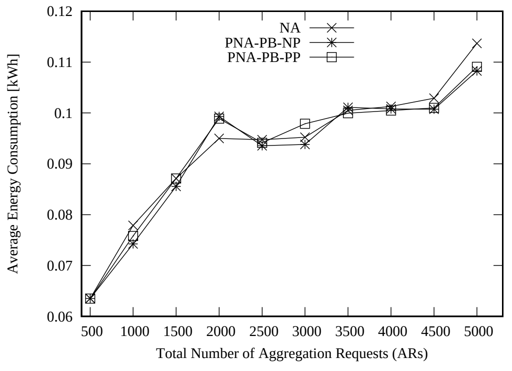
(a)



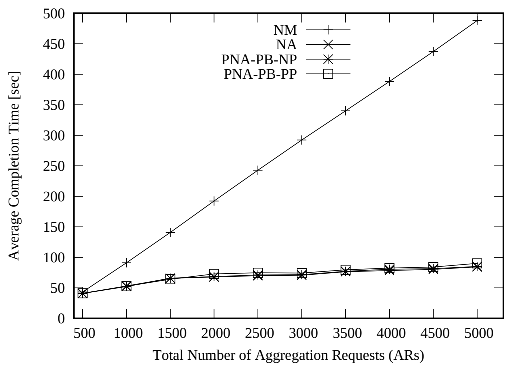
(b)



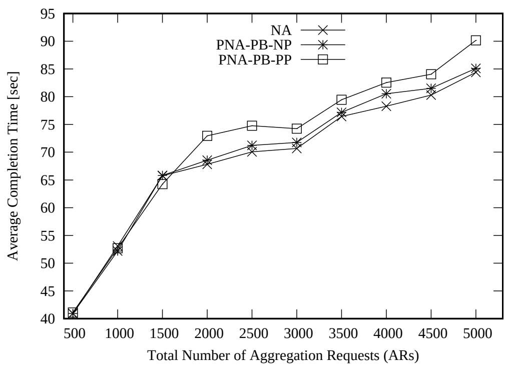
(c)



(d)



(e)



(f)

Figure 5.7: Comparing the efficiency of Network-Priority (NP) and Power-Priority (PP) final selection policies while Power Gain (PG) is calculated with Power Balancer (PB) method.

The three plots in the left column compare the NM approach with the three optimization approaches. Not surprisingly, the significant improvement is observed in terms of all the performance metrics. Nonetheless, it is difficult to distinguish between optimization approaches particularly for the average energy consumption and the average completion time. The three plots in the right column displays the difference between NA, PNA-PB-NP, and PNA-PB-PP approaches. The mean and 95% CI of average SLAV, average energy consumption, and average completion time presented in Figure 5.7 are illustrated in Appendix A, Tables A.1, A.2, and A.3, respectively.

Furthermore, t-test is used to compare the optimization approaches. The t-test results for the average SLAV, energy consumption, and completion time are illustrated in Appendix A, Tables A.4, A.5, and A.6, respectively. The aim of this comparison is to identify the power-network aware approach (either PNA-PB-NP or PNA-PB-PP) that increases the energy saving compared to NA approach. However, the average SLAV should not exceed that of NA approach. Three zones are discussed in this chapter: low load level from 500 to 2000 requests (zone 1), medium load level from 2500 to 3500 requests (zone 2), and high-load level from 4000 to 5000 requests (zone 3).

The t-test results can be summarized as follow: Firstly, Appendix A, Tables A.4 shows that the PNA-PB-NP approach does not have any negative effects on the average SLAV compared to the NA. Nonetheless, it can be observed that the SLAV in PNA-PB-PP is significantly higher than that of NA especially for high load conditions (zone 2 and 3).

Secondly, the same observation holds true for the average completion time in Appendix A, Tables A.6. While the average completion time in PNA-PB-PP in zone 2 and 3 are higher compared to that of NA, no statistical difference is observed between PNA-PB-NP and NA. Thirdly, Appendix A, Tables A.5 reveals that the average energy consumption in PNA-PB-NP is lower in zone 1 compared to that of NA. However, the difference of average energy consumption between PNA-PB-PP and NA is not statistically different. Hence, it is concluded that PNA-PB-NP is the winner of this comparison because it can save more energy than other approaches while its average SLAV and completion time are not higher than those of NA.

Stage 1 is continued by analyzing the effect of NP and PP policies on the performance metrics whilst PS method is used. Four approaches are studied: the No Migration (NM) approach, the network-aware approach (NA), the power-network aware approach with PS method and NP policy (PNA-PS-NP), and the power-network aware approach with PS method and PP policy (PNA-PS-PP). Figure 5.8 illustrates the simulation results. Obviously, the optimization approaches results in the significant improvement compared to NM approach (Figures 5.8(a), 5.8(c), and 5.8(e)). The mean and 95% CI of the average SLAV, average energy consumption, and average completion time are presented in Appendix A, Tables A.7, A.8, and A.9, respectively. The optimization approaches are compared using t-test.

The t-test results in terms of average SLAV, energy consumption and completion time are displayed in in Appendix A, Tables A.10, A.11, and A.12, respectively. In the following, a brief description of the results is given. First, Appendix A, Table A.10 and Appendix A, Table A.12 show that the average SLAV and the completion time in PNA-PS-NP and PNA-PS-PP approaches are significantly higher compared to that of NA when load increases (i.e. zone 2 and 3). However, no statistical differences in terms of average SLAV and completion time is observed in zone 1. Additionally, Appendix A, Table A.11 clearly shows that PNA-PS-NP consumes lower energy compared to NA in zone 1, but PNA-PS-PP and NA consume statistically similar energy in all zones. Thus, despite the better efficiency of PNA-PS-NP over PNA-PS-PP, the former is only preferable to NA in zone 1. From the results in stage 1, the author concludes that PNA-PB-NP approach is the optimal substitution for the NA approach in all zones.

In the second stage, the NP policy is adapted as the final selection policy and PB is compared with PS in more details. In particular, this stage determines the main parameters that contribute to the lower average completion time in PB compared to that of PS. The simulation results are illustrated in Figure 5.9. The PB and PS methods are represented by PNA-PB-NP and PNA-PS-NP, respectively. Furthermore, the

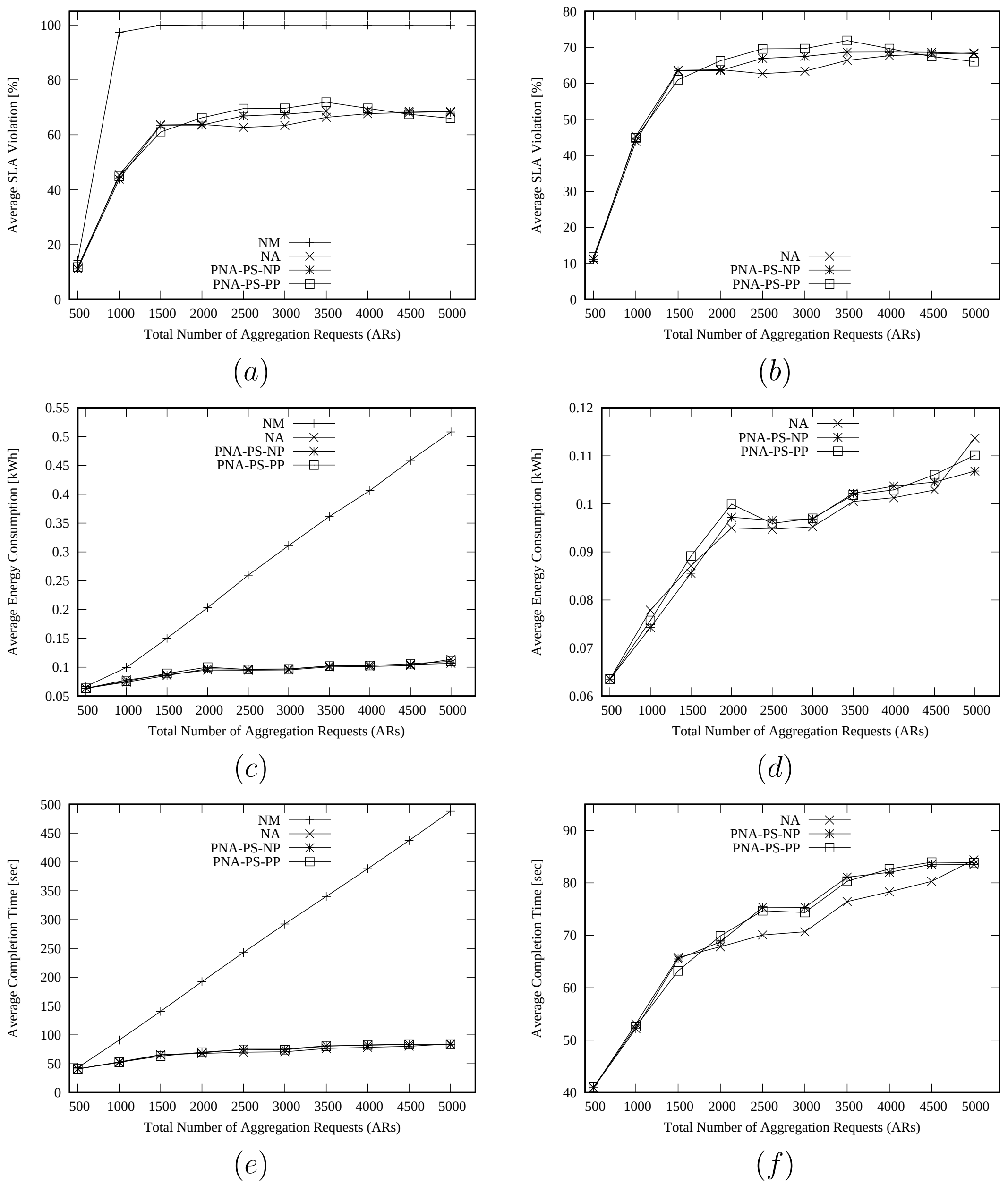


Figure 5.8: Comparing the efficiency of Network-Priority (NP) and Power-Priority (PP) final selection policies while Power Gain (PG) is calculated with Power Saver (PS) method.

mean and 95% CI of the values are given in Appendix A, Table A.13. Based on the results from the t-test (refer to Appendix A, Table A.14), the average completion time in PB is lower than that of PS in zone 2 with *p*-values of 0.000005, 0.000024, and 0.002791 for 2500, 3000, and 3500 requests, respectively. However, no significant difference is observed between PB and PS in the other two zones (i.e. zone 1 and 3).

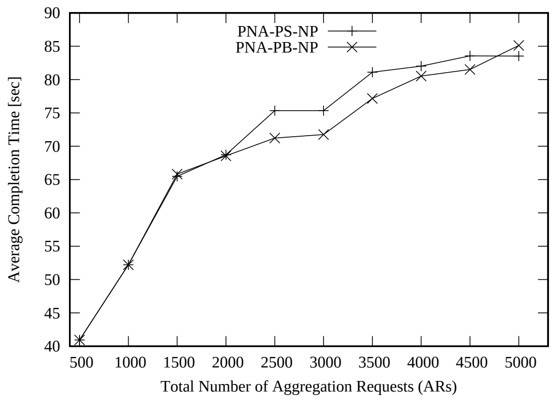


Figure 5.9: Comparing the average completion time of Power-Balancer (PB) and Power-Saver (PS) calculation methods. In this comparison the Network-Priority (NP) is adopted for the final selection policy.

Essentially, the average completion time is affected by two parameters: computing power and network bandwidth. When the system load increases, the CPU utilization grows and possibly exceeds the UUT. Certainly, this increases the average completion time because the highly loaded VMs receive less than their demanded CPU. Additionally, the total amount of steady traffic that is removed from the DCN has direct influence on the average completion time, and the influence becomes even more apparent when network-intensive applications are running.

Figure 5.10 illustrates the average traffic gain on PB and PS methods. The mean and 95% CI of the values can be seen in Appendix A, Table A.15. The t-test results (Appendix A, Table A.16) reveal that PB and PS are statistically similar in zone 1, but PB achieves higher traffic gain compared to that of PS starting from zone 2.

The average over-utilization time of PB and PS are depicted in Figure 5.11 (The mean and 95% CI of values are shown in Appendix A, Table A.17). The t-test results (Appendix A, Table A.18) demonstrates that the average over-utilization time between

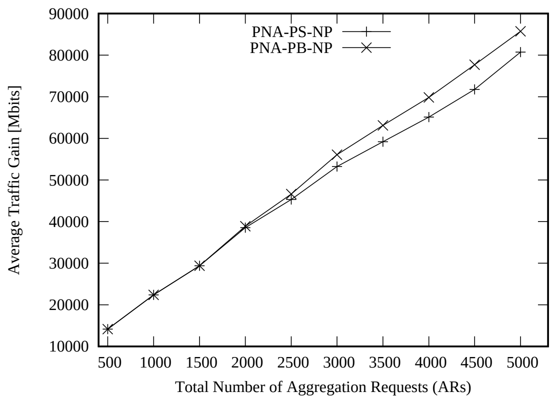


Figure 5.10: Comparing the average traffic gain of Power-Balancer (PB) and Power-Saver (PS) calculation methods. In this comparison the Network-Priority (NP) is adopted for the final selection policy.

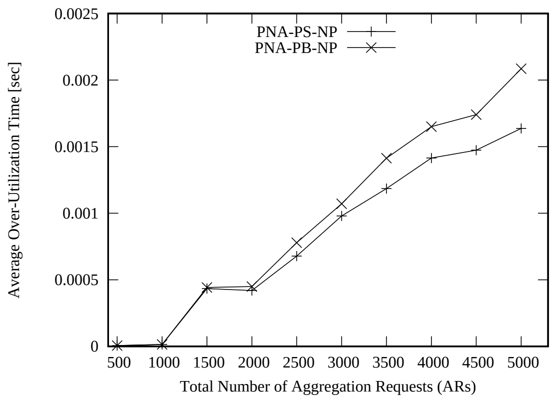


Figure 5.11: Comparing the average over-utilization time of Power-Balancer (PB) and Power-Saver (PS) calculation methods. In this comparison the Network-Priority (NP) is adopted for the final selection policy.

PB and PS are statistically similar in zone 1 and zone 2. Hence, it is concluded that the difference of average completion time between PB and PS is not very significant in zone 1, but it is statistically notable in zone 2 (see Figure 5.9) because while there is no difference between their average over-utilization time, PB can achieve higher traffic gain.

5.4 Summary

This chapter proposes an energy and network aware VM migration heuristic for three-tier Web applications running on the Cloud. The objectives of this chapter are to minimize the average SLA violation and energy consumption of a data center. This work consist of two parts. Firstly, a network-aware VM migration heuristic presented in the previous chapter is evaluated under a new system configuration consisting of three host models. The experimental results confirm that MST overloading detection policy is much more effective in terms of average SLA violation and energy consumption compared to MLB one.

Secondly, the proposed network-aware VM migration algorithm is extended to include energy-awareness capabilities. The new algorithm calculates NG and Power Gain (PG) for each candidate VM and creates NG and PG lists for each congested link. The PG is calculated using Power Balancer (PB) and Power Saver (PS) methods. Eventually, the VM with the lowest sum of ranks is selected for migration. In the case of multiple options with the same rank, two final selection policies are applied: Network Priority (NP) and Power Priority (PP). An extensive simulation study is conducted to determine the best integration of PG calculation methods and final selection policies. The simulation results indicate that integration of PB method with NP policy leads to the best performance.

Chapter 6

Conclusions and Future Works

This chapter summarizes the key findings and contributions of this thesis. Moreover, it presents several avenues for future research directions on VM migration in cloud.

6.1 Conclusions

The main contents of the thesis can be summarized as follows:

- **Chapter 1:** The fundamental background of cloud computing is provided and some applications of VM migration are presented.

- **Chapter 2:** The research works related to VM management algorithms are presented and the current trend and novelty of this dissertation is discussed.

- **Chapter 3:** In this chapter, *WPress* is proposed as an open-source performance benchmark for three-tier Web applications running on public clouds. *WPress* is tested for small and large VM instances of Amazon EC2, Microsoft Azure and Rackspace Cloud. Three different workload types (Read, Write and Read/Write) are considered. Furthermore, *WPressClient* is implemented as an open-source client and load generator application. WPressClient can generate representative workloads for WordPress and calculate the average response time and total cost of VMs for each provider. Moreover, the CPU micro-benchmark presented in [74] is employed to analyze CPU performance on the clouds under study and its impact on *WPress* results is identified.

The experimental results have shown that Rackspace Cloud and Microsoft Azure are the best cloud solutions for small and large VM instances, respectively. Furthermore, it is noticed that average response time has substantial fluctuations on large instances that are due to undesirable effect of CPU affinity and significant performance variation of CPU. This shows that an efficient resource optimization method is crucial.

- **Chapter 4:** This chapter proposed an adaptive network-aware migration algorithm with the objective of minimizing the SLA violation for three-tier Web applications. The high energy consumption of data centers is addressed by using MBFD power-aware VM placement algorithm proposed in [92]. In addition, CloudSim is extended by adding a new module that integrates the energy-efficiency of the power module [92] with the communication models introduced in the network module [109]. The new module is able to model three-tier web applications running on clouds and implements packet-switching transmission system. The new module enables researchers to simulate the various energy and network aware VM allocation and migration algorithm.

  Two host overloading detection polices are compared. First, the MST policy that sets a static upper utilization threshold for all hosts. Second, the MLB policy which considers the average load of the system as the upper utilization threshold. The experimental results and the statistical analysis clearly demonstrate that the network-aware migration algorithm coupled with either MST or MLB can meet the main objective of this research work by providing significant improvement in terms of SLA violation and energy consumption.

  The author wants to draw the reader's attention to the fact that MBFD algorithm is only applied for initial VM placement. The proposed network-aware heuristic identifies congested links, calculates the Network Gain (NG) for all candidate VMs, and migrates the one with the maximum NG, regardless of its effect on energy consumption.

- **Chapter 5:** This chapter proposes combined energy efficient and network aware VM migration heuristic for three-tier Web applications running on the cloud. The objectives of the heuristic is to minimize the average SLA violation and energy consumption of a data center. This work consist of two parts. Firstly, a network-aware VM migration heuristic presented in the Chapter 4, is evaluated under a new system configuration consisting of three host models. The experimental results confirm that MST overloading detection policy is much more effective in terms of average SLA violation and energy consumption compared to MLB one.

  Secondly, the proposed network-aware VM migration algorithm is extended to include energy-awareness capabilities. The new algorithm calculates NG and Power Gain (PG) for each candidate VM and creates NG and PG lists for each congested link. The PG is calculated using Power Balancer (PB) and Power Saver (PS) methods. Eventually, the VM with the lowest sum of ranks is selected for migration. In the case of multiple options with the same rank, two final selection policies are applied: Network Priority (NP) and Power Priority (PP). An extensive simulation study is conducted to determine the best integration of PG calculation methods and final selection policies. The simulation results indicate that integration of PB method with NP policy leads to the best performance.

6.2 Future Works

As stated in Chapter 2, several research works have been done in the field of benchmarking application for three-tier applications. However, designing a comprehensive benchmark, suitable for cloud environments, with unknown hardware configurations and pay-as-you-go pricing model, is a challenging task. Moreover, it is shown that energy efficient and network aware VM management policies have been receiving a lot of attention. Nonetheless, there are still some research problems that have not been well studied in this topic. This section presents some of the potential avenues for future research.

6.2.1 A Comprehensive Benchmark Application

In this thesis, the first steps are made towards designing a comprehensive benchmark suit for three-tier enterprise applications. The proposed benchmark suit consists of a single three-tier Web application and two performance metrics including average response time and average operational cost. In fact, this work is an effort to investigate the effect of CPU virtualization on the performance metrics. The author makes a few suggestions to improve the reliability of the proposed benchmark suit.

Firstly, it is very beneficial to incorporate more diverse set of applications in the benchmark suit including three-tier Web applications, High Performance Computing (HPC), e-commerce, and social networks [109]. This helps to have more accurate performance evaluation of the cloud under test. Another suggestion is to consider new metrics to evaluate disk and network performance of a cloud provider. These metrics are important because three-tier Web applications typically interact with millions of users. This makes a large amount of data to be stored on data bases and huge amount of traffic to be transmitted over the data center network.

6.2.2 Energy-Efficient Network-Aware VM Migration Algorithm

In this thesis, an energy-efficient network-aware VM migration heuristic is proposed for three-tier Web applications running on clouds. However, several assumptions are made without which the simulated system would be more realistic. The author offers the following suggestions towards approaching a comprehensive VM migration algorithm for three-tier Web applications in clouds.

6.2.2.1 Over-Loading Detection Policies

In this thesis, the MST [92] and MLB [141] overloading detection policies are compared and it is shown that MST results in lower SLA violation and energy consumption compared to those of MLB. Nonetheless, there are several adaptive overloading detection policies that can be studied as future work. One suggestion is to compare the adaptive overloading detection policies proposed in [121] including Median Absolute

Deviation (MAD), Inter Quartile Range (IQR), and Local Regression (LR) with MST. This helps to identify the most efficient overloading detection policy to be integrated with the proposed heuristics in terms of SLA violation and energy consumption.

6.2.2.2 Fault Tolerant VM Migration Algorithm

This thesis considers the effect of network congestion on the SLA violation and energy consumption of data centers on a fault-free system. However, there are several factors that have negative influences on the performance and energy consumption of clouds. One of the important factors is the failure occurrence in data centers. A failure may occur in the hardware layer including servers, power supplies, network links, switches, and routers. It also may happen in the software layer affecting the cloud hypervisor or the client's jobs.

Fault-tolerant approaches are mainly categorized in three groups including failure detection policies, failure rollover techniques, and failure prediction algorithms [149]. In particular, the VM migration approaches are very helpful in failure rollover and failure prediction algorithms [150]. Therefore, a good research direction is to integrate the proposed VM migration algorithm with new fault-tolerant capabilities.

6.2.2.3 Three-Tiered Data Center Network Architecture

The current work assumed a data center with a single edge switch. This is due to the fact that at the time of writing this thesis, it is one of the first works to implement an energy-efficient network-aware VM migration algorithm for three-tier applications in CloudSim. Hence, for the sake of simplicity, integration of complex routing algorithm with the proposed method is not considered. However, real data centers typically use three-tiered architecture including edge, aggregation, and core switches organized in several topologies such as tree, fat-tree, and VL2 [105]. Hence, one direction for future work is to consider a complete data center and find the most efficient routing algorithm to be integrated with the proposed heuristics.

6.2.2.4 VM Migration in WAN Environments

This thesis proposes several VM migration heuristics within a single data center. The concept of VM migration algorithm in the WAN environment is an interesting research topic that has been discussed in previous literature [115, 151, 152]. On one hand, in the WAN environment, the high bandwidth over-subscription causes limited available bandwidth between data centers. On the other hand, VM migration is known to be a costly operation in terms of network bandwidth and CPU utilization [153]. These may impose new constraints on the VM selection policies and the destination selection methods proposed in this thesis.

6.2.3 Resource Management for HPC-type Applications

This dissertation has mainly studied the resource management problem for three-tier Web applications running on cloud. In fact, cloud data centers are able to provide massive computing resources. This makes cloud computing attractive for High Performance Computing (HPC) applications [154–156]. An interesting research avenue includes modifying the proposed energy-efficient network-aware VM migration algorithm for HPC applications running on cloud. As an example, readers are encouraged to study how the proposed approach can be used in the genomic data analysis.

6.2.3.1 Genomic Data Analysis in Cloud

Genomics is used to study the similarities and differences of the entire mammalian genome [157,158] which encompasses sequencing, mapping and analyzing an extensive range of Deoxyribonucleic Acid (DNA) and Ribonucleic Acid (RNA) codes [159]. Introduction of modern data collection approaches has reduced the cost of genomic data collection which had been a big challenge historically. Nonetheless, handling and analyzing the substantial amount of genomic data requires powerful systems in terms of computing power, network bandwidth and storage capacity. To address this issue, cloud computing has been introduced as a substitution for high performance computing (HPC) systems [160].

Apparently, analyzing the huge amount of genomic data stored on the distributed data centers generates significant traffic on the network. An interesting area for future study is to integrate the VM migration heuristics proposed in this thesis for the genomic analysis applications.

Appendix A

The Statistical Analysis of Section 5.3.4

This appendix shows the mean, confidence interval (CI), and t-test analysis of the experimental results depicted in Section 5.3.4.

Table A.1: The mean and 95% CI of NM, NA, PNA-PB-NP, and PNA-PB-PP for Average SLAV (%). The mean values are presented in Figure 5.7(a).

Requests	NM	NA	PNA-PB-NP	PNA-PB-PP
500	14.200	11.140	11.140	11.820
	(13.790,14.610)	(9.325,12.955)	(9.325,12.955)	(9.454,14.186)
1000	97.320	45.370	43.880	44.820
	(97.182,97.458)	(43.016,47.724)	(41.504,46.256)	(42.947,46.693)
1500	99.880	63.600	63.447	62.980
	(99.850,99.910)	(61.717,65.483)	(61.807,65.087)	(61.670,64.290)
2000	100.000	63.795	63.070	70.105
	(100.000,100.000)	(61.887,65.703)	(61.245,64.895)	(68.112,72.098)
2500	100.000	62.680	63.696	71.648
	(100.000,100.000)	(61.458,63.902)	(62.263,65.129)	(69.880,73.416)
3000	100.000	63.393	66.083	70.597
	(100.000,100.000)	(61.635,65.152)	(64.776,67.391)	(68.719,72.475)
3500	100.000	66.391	67.689	72.671
	(100.000,100.000)	(64.446,68.337)	(65.663,69.714)	(70.351,74.992)
4000	100.000	67.680	69.695	75.110
	(100.000,100.000)	(66.176,69.184)	(66.779,72.611)	(72.742,77.478)
4500	100.000	68.158	70.804	75.247
	(100.000,100.000)	(65.905,70.410)	(68.184,73.425)	(72.901,77.593)
5000	100.000	68.470	70.138	75.624
	(100.000,100.000)	(66.041,70.899)	(66.962,73.314)	(73.229,78.019)

Table A.2: The mean and 95% CI of NM, NA, PNA-PB-NP, and PNA-PB-PP for average energy consumption (kWh). The mean values are presented in Figure 5.7(c).

Requests	NM	NA	PNA-PB-NP	PNA-PB-PP
500	0.066	0.064	0.064	0.063
	(0.066,0.067)	(0.063,0.064)	(0.063,0.064)	(0.063,0.064)
1000	0.100	0.078	0.074	0.076
	(0.099,0.100)	(0.077,0.078)	(0.074,0.075)	(0.073,0.079)
1500	0.150	0.087	0.086	0.087
	(0.150,0.150)	(0.086,0.088)	(0.085,0.087)	(0.086,0.089)
2000	0.203	0.095	0.099	0.099
	(0.203,0.204)	(0.088,0.102)	(0.090,0.109)	(0.091,0.107)
2500	0.260	0.095	0.094	0.094
	(0.259,0.260)	(0.091,0.098)	(0.093,0.095)	(0.093,0.095)
3000	0.311	0.095	0.094	0.098
	(0.311,0.311)	(0.092,0.098)	(0.092,0.095)	(0.094,0.102)
3500	0.361	0.100	0.101	0.100
	(0.361,0.362)	(0.098,0.103)	(0.095,0.107)	(0.095,0.105)
4000	0.406	0.101	0.101	0.100
	(0.406,0.407)	(0.099,0.104)	(0.099,0.102)	(0.098,0.103)
4500	0.459	0.103	0.101	0.101
	(0.457,0.461)	(0.102,0.104)	(0.099,0.102)	(0.098,0.104)
5000	0.508	0.114	0.108	0.109
	(0.508,0.508)	(0.110,0.117)	(0.106,0.111)	(0.107,0.111)

Table A.3: The mean and 95% CI of NM, NA, PNA-PB-NP, and PNA-PB-PP for average completion time (sec). The mean values are presented in Figure 5.7(e).

Requests	NM	NA	PNA-PB-NP	PNA-PB-PP
500	43.274	40.931	40.931	41.110
	(43.170,43.378)	(40.355,41.508)	(40.355,41.508)	(40.444,41.777)
1000	91.067	53.093	52.203	52.699
	(90.926,91.207)	(52.218,53.968)	(51.359,53.047)	(52.135,53.263)
1500	140.929	65.744	65.819	64.259
	(140.728,141.129)	(64.438,67.049)	(64.609,67.028)	(63.002,65.515)
2000	192.198	67.830	68.564	72.939
	(191.989,192.406)	(66.262,69.397)	(66.933,70.194)	(71.742,74.136)
2500	242.849	70.070	71.227	74.777
	(242.481,243.217)	(68.733,71.408)	(70.118,72.335)	(73.608,75.947)
3000	292.233	70.668	71.748	74.251
	(291.934,292.532)	(68.898,72.437)	(70.693,72.802)	(73.155,75.348)
3500	340.080	76.440	77.166	79.451
	(339.611,340.550)	(74.845,78.034)	(75.643,78.688)	(77.773,81.129)
4000	388.311	78.276	80.528	82.535
	(387.682,388.940)	(76.286,80.266)	(78.638,82.417)	(80.372,84.699)
4500	437.373	80.301	81.509	84.056
	(436.754,437.992)	(77.738,82.864)	(79.201,83.817)	(81.385,86.727)
5000	487.955	84.395	85.118	90.166
	(487.581,488.328)	(81.843,86.947)	(82.737,87.499)	(87.994,92.338)

Table A.4: The t-test results to compare average SLAV (%) between NA, PNA-PB-NP, and PNA-PB-PP. The original values are presented in Figure 5.7(b).

Group 1 = PNA-PB-NP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000 (-2.384, 2.384)	1.000
1000	-1.490 (-4.597, 1.617)	0.326
1500	-0.153 (-2.472, 2.166)	0.891
2000	-0.725 (-3.177, 1.727)	0.542
2500	1.016 (-0.733, 2.765)	0.238
3000	2.690 (0.655, 4.725)	0.013
3500	1.297 (-1.311, 3.905)	0.309
4000	2.015 (-1.032, 5.062)	0.187
4500	2.647 (-0.563, 5.856)	0.100
5000	1.668 (-2.046, 5.382)	0.358
Group 1 = PNA-PB-PP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.680 (-2.089, 3.449)	0.612
1000	-0.550 (-3.344, 2.244)	0.684
1500	-0.620 (-2.750, 1.510)	0.549
2000	6.310 (3.747, 8.873)	$< .001$
2500	8.968 (6.972, 10.964)	$< .001$
3000	7.203 (4.814, 9.593)	$< .001$
3500	6.280 (3.468, 9.092)	$< .001$
4000	7.430 (4.825, 10.035)	$< .001$
4500	7.089 (4.068, 10.110)	$< .001$
5000	7.154 (3.986, 10.322)	$< .001$
Group 1 = PNA-PB-PP Group 2 = PNA-PB-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.680 (-2.089, 3.449)	0.612
1000	0.940 (-1.870, 3.750)	0.491
1500	-0.467 (-2.416, 1.482)	0.621
2000	7.035 (4.525, 9.545)	$< .001$
2500	7.952 (5.838, 10.066)	$< .001$
3000	4.513 (2.388, 6.639)	$< .001$
3500	4.983 (2.122, 7.844)	0.001
4000	5.415 (1.926, 8.904)	0.004
4500	4.442 (1.175, 7.709)	0.010
5000	5.486 (1.791, 9.181)	0.006

Table A.5: The t-test results to compare average energy consumption (kWh) between NA, PNA-PB-NP, and PNA-PB-PP. The original values are presented in Figure 5.7(d).

Group 1 = PNA-PB-NP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000 (-0.000, 0.000)	1.000
1000	-0.004 (-0.004, -0.003)	$< .001$
1500	-0.002 (-0.003, -0.000)	0.026
2000	0.004 (-0.007, 0.015)	0.422
2500	-0.001 (-0.004, 0.002)	0.470
3000	-0.001 (-0.005, 0.002)	0.402
3500	0.001 (-0.005, 0.006)	0.821
4000	-0.000 (-0.003, 0.002)	0.725
4500	-0.002 (-0.004, -0.001)	0.006
5000	-0.005 (-0.009, -0.002)	0.006
Group 1 = PNA-PB-PP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	-0.000 (-0.000, 0.000)	0.865
1000	-0.002 (-0.005, 0.001)	0.142
1500	0.000 (-0.002, 0.002)	0.982
2000	0.004 (-0.006, 0.014)	0.430
2500	-0.001 (-0.004, 0.003)	0.713
3000	0.003 (-0.002, 0.008)	0.282
3500	-0.001 (-0.005, 0.004)	0.821
4000	-0.001 (-0.004, 0.002)	0.602
4500	-0.002 (-0.005, 0.001)	0.172
5000	-0.005 (-0.008, -0.001)	0.015
Group 1 = PNA-PB-PP Group 2 = PNA-PB-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	-0.000 (-0.000, 0.000)	0.865
1000	0.002 (-0.001, 0.004)	0.256
1500	0.002 (-0.000, 0.003)	0.054
2000	-0.000 (-0.012, 0.011)	0.943
2500	0.001 (-0.001, 0.002)	0.355
3000	0.004 (-0.000, 0.008)	0.070
3500	-0.001 (-0.008, 0.006)	0.729
4000	-0.000 (-0.003, 0.002)	0.782
4500	0.000 (-0.003, 0.003)	0.840
5000	0.001 (-0.002, 0.004)	0.567

Table A.6: The t-test results to compare average completion time (sec) between NA, PNA-PB-NP, and PNA-PB-PP. The original values are presented in Figure 5.7(e).

Group 1 = PNA-PB-NP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000 (-0.757, 0.757)	1.000
1000	-0.890 (-2.019, 0.239)	0.114
1500	0.075 (-1.578, 1.728)	0.925
2000	0.734 (-1.367, 2.834)	0.472
2500	1.156 (-0.457, 2.769)	0.150
3000	1.080 (-0.833, 2.993)	0.254
3500	0.726 (-1.322, 2.774)	0.466
4000	2.252 (-0.296, 4.801)	0.079
4500	1.208 (-1.996, 4.411)	0.438
5000	0.723 (-2.519, 3.964)	0.645
Group 1 = PNA-PB-PP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.179 (-0.639, 0.997)	0.651
1000	-0.394 (-1.360, 0.573)	0.405
1500	-1.485 (-3.168, 0.198)	0.080
2000	5.109 (3.277, 6.941)	< 0.001
2500	4.707 (3.057, 6.357)	< 0.001
3000	3.583 (1.650, 5.517)	0.001
3500	3.011 (0.861, 5.161)	0.008
4000	4.260 (1.529, 6.990)	0.004
4500	3.755 (0.317, 7.193)	0.034
5000	5.771 (2.658, 8.883)	0.001
Group 1 = PNA-PB-PP Group 2 = PNA-PB-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.179 (-0.639, 0.997)	0.651
1000	0.497 (-0.446, 1.439)	0.285
1500	-1.560 (-3.179, 0.060)	0.058
2000	4.375 (2.497, 6.253)	< 0.001
2500	3.551 (2.054, 5.047)	< 0.001
3000	2.504 (1.090, 3.917)	0.001
3500	2.285 (0.181, 4.389)	0.035
4000	2.007 (-0.661, 4.675)	0.131
4500	2.547 (-0.731, 5.826)	0.120
5000	5.048 (2.055, 8.041)	0.002

Table A.7: The mean and 95% CI of NM, NA, PNA-PS-NP, and PNA-PS-PP for Average SLAV (%). The mean values are presented in Figure 5.8(a).

Requests	NM	NA	PNA-PS-NP	PNA-PS-PP
500	14.200	11.140	11.140	11.800
	(13.790,14.610)	(9.325,12.955)	(9.325,12.955)	(9.472,14.128)
1000	97.320	45.370	43.880	44.870
	(97.182,97.458)	(43.016,47.724)	(41.504,46.256)	(43.155,46.585)
1500	99.880	63.600	63.487	61.007
	(99.850,99.910)	(61.717,65.483)	(61.830,65.143)	(60.075,61.938)
2000	100.000	63.795	63.575	66.250
	(100.000,100.000)	(61.887,65.703)	(61.811,65.339)	(64.865,67.635)
2500	100.000	62.680	66.904	69.584
	(100.000,100.000)	(61.458,63.902)	(65.360,68.448)	(69.016,70.152)
3000	100.000	63.393	67.503	69.650
	(100.000,100.000)	(61.635,65.152)	(66.346,68.660)	(68.348,70.952)
3500	100.000	66.391	68.666	71.877
	(100.000,100.000)	(64.446,68.337)	(67.431,69.900)	(70.496,73.258)
4000	100.000	67.680	68.710	69.640
	(100.000,100.000)	(66.176,69.184)	(67.167,70.253)	(68.041,71.239)
4500	100.000	68.158	68.598	67.480
	(100.000,100.000)	(65.905,70.410)	(67.315,69.880)	(66.247,68.713)
5000	100.000	68.470	68.258	66.022
	(100.000,100.000)	(66.041,70.899)	(66.668,69.848)	(64.905,67.139)

Table A.8: The mean and 95% CI of NM, NA, PNA-PS-NP, and PNA-PS-PP for average energy consumption (kWh). The mean values are presented in Figure 5.8(c).

Requests	NM	NA	PNA-PS-NP	PNA-PS-PP
500	0.066	0.064	0.064	0.064
	(0.066,0.067)	(0.063,0.064)	(0.063,0.064)	(0.063,0.064)
1000	0.100	0.078	0.074	0.076
	(0.099,0.100)	(0.077,0.078)	(0.074,0.075)	(0.073,0.079)
1500	0.150	0.087	0.086	0.089
	(0.150,0.150)	(0.086,0.088)	(0.084,0.087)	(0.088,0.090)
2000	0.203	0.095	0.097	0.100
	(0.203,0.204)	(0.088,0.102)	(0.088,0.106)	(0.092,0.108)
2500	0.260	0.095	0.097	0.096
	(0.259,0.260)	(0.091,0.098)	(0.096,0.098)	(0.093,0.099)
3000	0.311	0.095	0.097	0.097
	(0.311,0.311)	(0.092,0.098)	(0.095,0.098)	(0.094,0.100)
3500	0.361	0.100	0.102	0.102
	(0.361,0.362)	(0.098,0.103)	(0.100,0.105)	(0.098,0.106)
4000	0.406	0.101	0.104	0.103
	(0.406,0.407)	(0.099,0.104)	(0.102,0.105)	(0.101,0.105)
4500	0.459	0.103	0.105	0.106
	(0.457,0.461)	(0.102,0.104)	(0.103,0.107)	(0.103,0.109)
5000	0.508	0.114	0.107	0.110
	(0.508,0.508)	(0.110,0.117)	(0.103,0.110)	(0.106,0.115)

Table A.9: The mean and 95% CI of NM, NA, PNA-PS-NP, and PNA-PS-PP for average completion time (sec). The mean values are presented in Figure 5.8(e).

Requests	NM	NA	PNA-PS-NP	PNA-PS-PP
500	43.274	40.931	40.931	41.035
	(43.170,43.378)	(40.355,41.508)	(40.355,41.508)	(40.375,41.696)
1000	91.067	53.093	52.203	52.555
	(90.926,91.207)	(52.218,53.968)	(51.359,53.047)	(52.027,53.084)
1500	140.929	65.744	65.478	63.229
	(140.728,141.129)	(64.438,67.049)	(64.133,66.823)	(62.570,63.887)
2000	192.198	67.830	68.734	69.890
	(191.989,192.406)	(66.262,69.397)	(67.875,69.594)	(68.843,70.937)
2500	242.849	70.070	75.346	74.704
	(242.481,243.217)	(68.733,71.408)	(74.463,76.229)	(74.233,75.175)
3000	292.233	70.668	75.336	74.346
	(291.934,292.532)	(68.898,72.437)	(74.360,76.313)	(73.568,75.125)
3500	340.080	76.440	81.104	80.330
	(339.611,340.550)	(74.845,78.034)	(79.067,83.141)	(78.481,82.178)
4000	388.311	78.276	82.016	82.683
	(387.682,388.940)	(76.286,80.266)	(80.806,83.226)	(80.852,84.515)
4500	437.373	80.301	83.554	83.942
	(436.754,437.992)	(77.738,82.864)	(82.043,85.065)	(82.207,85.678)
5000	487.955	84.395	83.535	83.873
	(487.581,488.328)	(81.843,86.947)	(81.096,85.973)	(82.459,85.287)

Table A.10: The t-test results to compare average SLAV (%) between NA, PNA-PS-NP, and PNA-PS-PP. The original values are presented in Figure 5.8(b).

Group 1 = PNA-PS-NP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000 (-2.384, 2.384)	1.000
1000	-1.490 (-4.597, 1.617)	0.326
1500	-0.113 (-2.443, 2.216)	0.919
2000	-0.220 (-2.633, 2.193)	0.850
2500	4.224 (2.395, 6.053)	< 0.001
3000	4.110 (2.155, 6.065)	< 0.001
3500	2.274 (0.135, 4.414)	0.040
4000	1.030 (-0.972, 3.032)	0.293
4500	0.440 (-1.968, 2.848)	0.706
5000	-0.212 (-2.908, 2.484)	0.870
Group 1 = PNA-PS-PP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.660 (-2.081, 3.401)	0.619
1000	-0.500 (-3.205, 2.205)	0.702
1500	-2.593 (-4.545, -0.642)	0.015
2000	2.455 (0.265, 4.645)	0.031
2500	6.904 (5.652, 8.156)	< 0.001
3000	6.257 (4.224, 8.289)	< 0.001
3500	5.486 (3.270, 7.701)	< 0.001
4000	1.960 (-0.079, 3.999)	0.058
4500	-0.678 (-3.063, 1.707)	0.560
5000	-2.448 (-4.931, 0.035)	0.059
Group 1 = PNA-PS-PP Group 2 = PNA-PS-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.660 (-2.081, 3.401)	0.619
1000	0.990 (-1.732, 3.712)	0.455
1500	-2.480 (-4.245, -0.715)	0.010
2000	2.675 (0.592, 4.758)	0.015
2500	2.680 (1.152, 4.208)	0.003
3000	2.147 (0.529, 3.765)	0.012
3500	3.211 (1.491, 4.932)	0.001
4000	0.930 (-1.134, 2.994)	0.356
4500	-1.118 (-2.770, 0.534)	0.172
5000	-2.236 (-4.041, -0.431)	0.019

Table A.11: The t-test results to compare average energy consumption (kWh) between NA, PNA-PS-NP, and PNA-PS-PP. The original values are presented in Figure 5.8(d).

Group 1 = PNA-PS-NP Group 2 = NA		
Requests	Difference (means and 95% CIs)	p-value
500	0.000 (-0.000, 0.000)	1.000
1000	-0.004 (-0.004, -0.003)	< 0.001
1500	-0.002 (-0.003, -0.000)	0.049
2000	0.002 (-0.008, 0.013)	0.660
2500	0.002 (-0.001, 0.005)	0.262
3000	0.002 (-0.002, 0.005)	0.323
3500	0.002 (-0.001, 0.005)	0.235
4000	0.002 (-0.000, 0.005)	0.097
4500	0.002 (-0.000, 0.004)	0.107
5000	-0.007 (-0.011, -0.002)	0.004
Group 1 = PNA-PS-PP Group 2 = NA		
Requests	Difference (means and 95% CIs)	p-value
500	-0.000 (-0.000, 0.000)	0.993
1000	-0.002 (-0.005, 0.001)	0.123
1500	0.002 (0.001, 0.003)	0.003
2000	0.005 (-0.005, 0.015)	0.317
2500	0.001 (-0.003, 0.005)	0.559
3000	0.002 (-0.003, 0.006)	0.400
3500	0.001 (-0.003, 0.006)	0.518
4000	0.002 (-0.002, 0.005)	0.301
4500	0.003 (0.000, 0.006)	0.054
5000	-0.004 (-0.009, 0.002)	0.162
Group 1 = PNA-PS-PP Group 2 = PNA-PS-NP		
Requests	Difference (means and 95% CIs)	p-value
500	-0.000 (-0.000, 0.000)	0.993
1000	0.001 (-0.001, 0.004)	0.275
1500	0.004 (0.002, 0.005)	< 0.001
2000	0.003 (-0.008, 0.014)	0.615
2500	-0.001 (-0.004, 0.002)	0.668
3000	0.000 (-0.003, 0.003)	0.928
3500	-0.000 (-0.005, 0.004)	0.871
4000	-0.001 (-0.003, 0.002)	0.511
4500	0.002 (-0.002, 0.005)	0.379
5000	0.003 (-0.002, 0.009)	0.203

Table A.12: The t-test results to compare average completion time (sec) between NA, PNA-PS-NP, and PNA-PS-PP. The original values are presented in Figure 5.8(e).

Group 1 = PNA-PS-NP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000 (-0.757, 0.757)	1.000
1000	-0.890 (-2.019, 0.239)	0.114
1500	-0.266 (-2.007, 1.475)	0.751
2000	0.905 (-0.756, 2.565)	0.271
2500	5.276 (3.787, 6.764)	< 0.001
3000	4.669 (2.792, 6.546)	< 0.001
3500	4.665 (2.262, 7.067)	< 0.001
4000	3.740 (1.577, 5.903)	0.002
4500	3.253 (0.490, 6.016)	0.026
5000	-0.860 (-4.139, 2.418)	0.588
Group 1 = PNA-PS-PP Group 2 = NA		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.104 (-0.711, 0.918)	0.792
1000	-0.538 (-1.487, 0.411)	0.252
1500	-2.515 (-3.873, -1.157)	0.001
2000	2.060 (0.310, 3.811)	0.025
2500	4.633 (3.317, 5.950)	< 0.001
3000	3.678 (1.883, 5.474)	< 0.001
3500	3.890 (1.623, 6.157)	0.002
4000	4.408 (1.896, 6.920)	0.001
4500	3.641 (0.767, 6.516)	0.017
5000	-0.522 (-3.232, 2.188)	0.691
Group 1 = PNA-PS-PP Group 2 = PNA-PS-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.104 (-0.711, 0.918)	0.792
1000	0.352 (-0.572, 1.277)	0.435
1500	-2.249 (-3.640, -0.858)	0.004
2000	1.155 (-0.103, 2.414)	0.070
2500	-0.642 (-1.572, 0.287)	0.169
3000	-0.990 (-2.150, 0.170)	0.090
3500	-0.775 (-3.329, 1.780)	0.532
4000	0.668 (-1.371, 2.706)	0.501
4500	0.388 (-1.749, 2.525)	0.707
5000	0.338 (-2.280, 2.957)	0.789

Table A.13: The mean and 95% CI of PNA-PB-NP and PNA-PS-NP in terms of average completion time (sec). The mean values are presented in Figure 5.9

Requests	PNA-PB-NP	PNA-PS-NP
500	40.931 (40.355,41.508)	40.931 (40.355,41.508)
1000	52.203 (51.359,53.047)	52.203 (51.359,53.047)
1500	65.819 (64.609,67.028)	65.478 (64.133,66.823)
2000	68.564 (66.933,70.194)	68.734 (67.875,69.594)
2500	71.227 (70.118,72.335)	75.346 (74.463,76.229)
3000	71.748 (70.693,72.802)	75.336 (74.360,76.313)
3500	77.166 (75.643,78.688)	81.104 (79.067,83.141)
4000	80.528 (78.638,82.417)	82.016 (80.806,83.226)
4500	81.509 (79.201,83.817)	83.554 (82.043,85.065)
5000	85.118 (82.737,87.499)	83.535 (81.096,85.973)

Table A.14: The t-test results to compare PNA-PB-NP and PNA-PS-NP in terms of average completion time (sec). The original values are presented in Figure 5.9

Group 1 = PNA-PS-NP		
Group 2 = PNA-PB-NP		
Requests	Difference (means and 95% CIs)	p-value
500	0.000 (-0.757, 0.757)	1.000
1000	0.000 (-1.109, 1.109)	1.000
1500	-0.341 (-2.021, 1.339)	0.674
2000	0.171 (-1.541, 1.883)	0.836
2500	4.119 (2.803, 5.436)	< 0.001
3000	3.589 (2.254, 4.923)	< 0.001
3500	3.939 (1.577, 6.300)	0.002
4000	1.488 (-0.596, 3.572)	0.153
4500	2.045 (-0.517, 4.607)	0.113
5000	-1.583 (-4.749, 1.582)	0.307

Table A.15: The mean and 95% CI of PNA-PB-NP and PNA-PS-NP in terms of average traffic gain (Mbits). The mean values are presented in Figures 5.10

Requests	PNA-PB-NP	PNA-PS-NP
500	14148.352 (13962.940,14333.764)	14148.352 (13962.940,14333.764)
1000	22367.114 (21782.328,22951.901)	22367.114 (21782.328,22951.901)
1500	29390.157 (28827.500,29952.815)	29390.080 (28564.989,30215.172)
2000	38889.888 (37916.771,39863.004)	38527.697 (37598.077,39457.317)
2500	46626.286 (45836.414,47416.158)	45313.417 (44230.253,46396.581)
3000	56073.872 (54844.466,57303.278)	53228.263 (52061.448,54395.078)
3500	63102.446 (62159.925,64044.967)	59189.998 (58060.424,60319.571)
4000	69828.285 (68374.022,71282.549)	65120.141 (64146.145,66094.136)
4500	77709.765 (76197.772,79221.758)	71771.688 (70640.980,72902.396)
5000	85739.243 (83788.207,87690.278)	80714.980 (79652.475,81777.485)

Table A.16: The t-test results to compare PNA-PB-NP and PNA-PS-NP in terms of average traffic gain (Mbits). The original values are presented in Figures 5.10

Group 1 = PNA-PS-NP		
Group 2 = PNA-PB-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000 (-243.532, 243.532)	1.000
1000	0.000 (-768.096, 768.096)	1.000
1500	-0.077 (-927.609, 927.456)	0.999
2000	-362.191 (-1612.105, 887.724)	0.550
2500	-1312.869 (-2557.941, -67.797)	0.041
3000	-2845.609 (-4419.821, -1271.397)	0.001
3500	-3912.448 (-5278.793, -2546.104)	< 0.001
4000	-4708.145 (-6333.752, -3082.537)	< 0.001
4500	-5938.077 (-7691.595, -4184.560)	< 0.001
5000	-5024.263 (-7087.584, -2960.941)	< 0.001

Table A.17: The mean and 95% CI of PNA-PB-NP and PNA-PS-NP in terms of average over-utilization time (sec). The mean values are presented in Figures 5.11

Requests	PNA-PB-NP	PNA-PS-NP
500	0.000006 (0.000001,0.000010)	0.000006 (0.000001,0.000010)
1000	0.000015 (0.000006,0.000023)	0.000015 (0.000006,0.000023)
1500	0.000442 (0.000404,0.000481)	0.000434 (0.000384,0.000485)
2000	0.000449 (0.000375,0.000523)	0.000420 (0.000347,0.000492)
2500	0.000779 (0.000732,0.000826)	0.000678 (0.000636,0.000719)
3000	0.001071 (0.000981,0.001161)	0.000979 (0.000885,0.001074)
3500	0.001413 (0.001288,0.001539)	0.001185 (0.001099,0.001270)
4000	0.001651 (0.001492,0.001810)	0.001414 (0.001308,0.001521)
4500	0.001740 (0.001597,0.001884)	0.001474 (0.001393,0.001555)
5000	0.002085 (0.001917,0.002253)	0.001637 (0.001489,0.001785)

Table A.18: The t-test results to compare PNA-PB-NP and PNA-PS-NP in terms of average over-utilization time (sec). The original values are presented in Figures 5.11

Group 1 = PNA-PS-NP		
Group 2 = PNA-PB-NP		
Requests	Difference (means and 95% CIs)	*p*-value
500	0.000000 (-0.000006, 0.000006)	1.000
1000	0.000000 (-0.000011, 0.000011)	1.000
1500	-0.000008 (-0.000067, 0.000051)	0.775
2000	-0.000029 (-0.000126, 0.000067)	0.528
2500	-0.000101 (-0.000159, -0.000043)	0.001
3000	-0.000092 (-0.000213, 0.000030)	0.131
3500	-0.000228 (-0.000370, -0.000087)	0.003
4000	-0.000236 (-0.000414, -0.000058)	0.013
4500	-0.000267 (-0.000420, -0.000114)	0.002
5000	-0.000448 (-0.000656, -0.000240)	< 0.001

Author's Publications

[1] **Amir Hossein Borhani**, Terence Hung, Bu-Sung Lee, and Zheng Qin, "Power-network aware VM migration heuristics for Multi-tier web applications", *Cluster Computing*, submitted.

[2] **Amir Hossein Borhani**, Terence Hung, Bu-Sung Lee, Zheng Qin, Zahra Bagheri, "Network-Aware VM Migration Heuristics for Improving the SLA Violation of Multi-Tier Web Applications in the Cloud", *IEEE 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 454-462, St. Petersburg, Russia, March, 2017.

[3] **Amir Hossein Borhani**, Philipp Leitner, Bu-Sung Lee, Xiaorong Li, Terence Hung, "WPress: An application-driven performance benchmark for cloud-based virtual machines", *IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC)*, pp. 101-109, Ulm, Germany, September, 2014.

References

[1] M.-T. Chen, C.-C. Hsu, M.-S. Kuo, Y.-J. Cheng, and C.-F. Chou, “GreenGlue: Power optimization for data centers through resource-guaranteed VM placement,” in *Proceedings of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom), and IEEE Cyber, Physical and Social Computing (CPSCom).* IEEE, 2014, pp. 510–517.

[2] P. G. J. Leelipushpam and J. Sharmila, “Live vm migration techniques in cloud environment - A survey,” in *Proceedings of the IEEE Conference on Information & Communication Technologies.* IEEE, 2013, pp. 408–413.

[3] L. Columbus. Roundup Of Cloud Computing Forecasts And Market Estimates, 2016. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2016/03/13/roundup-of-cloud-computing-forecasts-and-market-estimates-2016

[4] J. Geelan *et al.*, “Twenty-one experts define cloud computing,” *Cloud Computing Journal*, vol. 4, pp. 1–5, 2009.

[5] E. Knorr and G. Gruman, “What cloud computing really means,” *InfoWorld*, vol. 7, pp. 20–20, 2008.

[6] R. Buyya, C. S. Yeo, and S. Venugopal, “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities,” in *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications.* IEEE, 2008, pp. 5–13.

[7] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[8] R. L. Grossman, "The case for cloud computing," *IT professional*, vol. 11, no. 2, pp. 23–27, 2009.

[9] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *Proceedings of the 10th IEEE/ACM International Conference on Grid Computing.* IEEE, 2009, pp. 17–25.

[10] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proceedings of the 5th International Joint Conference on INC, IMS and IDC.* IEEE, 2009, pp. 44–51.

[11] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[12] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 311–336, 2011.

[13] S. Patidar, D. Rane, and P. Jain, "A survey paper on cloud computing," in *Proceedings of the 2nd International Conference on Advanced Computing & Communication Technologies.* IEEE, 2012, pp. 394–398.

[14] U. Moghe, P. Lakkadwala, and D. K. Mishra, "Cloud computing: Survey of different utilization techniques," in *Proceedings of the CSI 6th International conference on Software engineering.* IEEE, 2012, pp. 1–4.

[15] S. Bera, S. Misra, and J. J. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1477–1494, 2015.

[16] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

[17] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," vol. 37, no. 5, pp. 164–177, 2003.

[19] C. A. Waldspurger, "Memory resource management in vmware esx server," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 181–194, 2002.

[20] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Proceedings of the Linux symposium*, vol. 1, 2007, pp. 225–230.

[21] D. Adami, B. Martini, M. Gharbaoui, P. Castoldi, G. Antichi, and S. Giordano, "Effective resource control strategies using openflow in cloud data center," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2013, pp. 568–574.

[22] C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1021–1036, 2012.

[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[24] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proceedings of the 2nd International Workshop on Testing Database Systems*. ACM, 2009, pp. 9:1–9:6.

[25] M. B. Qureshi, M. M. Dehnavi, N. Min-Allah, M. S. Qureshi, H. Hussain, I. Rentifis, N. Tziritas, T. Loukopoulos, S. U. Khan, C.-Z. Xu *et al.*, "Survey on grid resource allocation mechanisms," *Journal of Grid Computing*, vol. 12, no. 2, pp. 399–441, 2014.

[26] G. Galante and L. C. E. de Bona, "A survey on cloud computing elasticity," in *Proceedings of the 5th International Conference on Utility and Cloud Computing*. IEEE, 2012, pp. 263–270.

[27] D. Sullivan. IaaS Providers List: 2014 Comparison And Guide - Tom's IT Pro. [Online]. Available: http://www.tomsitpro.com/articles/iaas-providers,1-1560.html

[28] ——. Paas providers list: Comparison and guide. [Online]. Available: http://www.tomsitpro.com/articles/paas-providers,1-1517.html

[29] L. Wu, S. K. Garg, and R. Buyya, “Sla-based resource allocation for software as a service provider (saas) in cloud computing environments,” in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* IEEE, 2011, pp. 195–204.

[30] H. Jin, S. Ibrahim, T. Bell, W. Gao, D. Huang, and S. Wu, “Cloud types and services,” in *Handbook of Cloud Computing.* Springer, 2010, pp. 335–355.

[31] S. Khoshnevis and F. Rabeifar, “Toward knowledge management as a service in cloud-based environments,” *International Journal of Mechatronics, Electrical and Computer Technology*, vol. 2, no. 4, pp. 88–110, 2012.

[32] Helena. Top 10 software as a service (saas) companies. [Online]. Available: http://zeendo.com/info/top-10-software-as-a-service-saas-companies/

[33] RightScale Ltd, “State of the cloud report,” December 2016.

[34] K. Xiong and H. Perros, “Service performance and analysis in cloud computing,” in *Proceedings of the 4th IEEE World Congress on Services.* IEEE, 2009, pp. 693–700.

[35] A. Alzahrani, N. Alalwan, and M. Sarrab, “Mobile cloud computing: advantage, disadvantage and open challenge,” in *Proceedings of the 7th Euro American Conference on Telematics and Information Systems.* ACM, 2014, p. 21.

[36] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, “Cloud computing—the business perspective,” *Decision support systems*, vol. 51, no. 1, pp. 176–189, 2011.

[37] AWS — Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. [Online]. Available: https://aws.amazon.com/ec2/

[38] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, “Microsoft azure and cloud computing,” in *Microsoft Azure.* Springer, 2015, pp. 3–26.

[39] R. Kumar, K. Jain, H. Maharwal, N. Jain, and A. Dadhich, “Apache cloudstack: Open source infrastructure as a service cloud computing platform,” *Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science*, pp. 111–116, 2014.

[40] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “Openstack: toward an open-source solution for cloud computing,” *International Journal of Computer Applications*, vol. 55, no. 3, 2012.

[41] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The eucalyptus open-source cloud-computing system,” in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid.* IEEE Computer Society, 2009, pp. 124–131.

[42] G. Singh, G. Garg, P. Jain, and H. Singh, “The structure of cloud engineering,” *International Journal of Computer Applications (0975–8887)*, vol. 33, no. 8, 2011.

[43] Google Cloud Platform. [Online]. Available: https://appengine.google.com/

[44] Amazon Web Services (AWS). [Online]. Available: https://aws.amazon.com/

[45] E. Arzuaga, “Using live virtual machine migration to improve resource efficiency in virtualized data centers,” Ph.D. dissertation, Department of Electrical and Computer Engineering, The Northeastern University, 2012.

[46] R. P. Goldberg, “Survey of virtual machine research,” *Computer*, vol. 7, no. 6, pp. 34–45, 1974.

[47] Y. Xing and Y. Zhan, “Virtualization and cloud computing,” in *Future Wireless Networks and Information Systems.* Springer, 2012, pp. 305–312.

[48] H. F. Cervone, “An overview of virtual and cloud computing,” *OCLC Systems & Services: International digital library perspectives*, vol. 26, no. 3, pp. 162–165, 2010.

[49] C. Rong, S. T. Nguyen, and M. G. Jaatun, “Beyond lightning: A survey on security challenges in cloud computing,” *Computers & Electrical Engineering*, vol. 39, no. 1, pp. 47–54, 2013.

[50] M. T. Jones. Cloud computing with Linux. [Online]. Available: https://www.ibm.com/developerworks/linux/library/l-cloud-computing/

[51] B. Furht, “Cloud computing fundamentals,” in *Handbook of cloud computing.* Springer, 2010, pp. 3–19.

[52] C. Xu, Z. Zhao, H. Wang, R. Shea, and J. Liu, "Energy efficiency of cloud virtual machines: From traffic pattern and cpu affinity perspectives," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–11, 2015.

[53] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 5, pp. 2–13, 2006.

[54] J. Fisher-Ogden, "Hardware support for efficient virtualization," *University of California, San Diego, Tech. Rep*, 2006.

[55] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "Vmware distributed resource management: Design, implementation, and lessons learned," *VMware Technical Journal*, vol. 1, no. 1, pp. 45–64, 2012.

[56] vCloud Air Infrastructure as a Service (IaaS) Hybrid Cloud. [Online]. Available: http://www.vmware.com/cloud-services/infrastructure.html

[57] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito, "Exploring container virtualization in iot clouds," in *Proceedings of the 2nd IEEE International Conference on Smart Computing.* IEEE, 2016, pp. 1–6.

[58] M. Rouse. What is containerization (container-based virtualization)? - definition from whatis.com. [Online]. Available: http://searchservervirtualization.techtarget.com/definition/container-based-virtualization-operating-system-level-virtualization

[59] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing.* IEEE, 2013, pp. 233–240.

[60] R. Shea, H. Wang, and J. Liu, "Power consumption of virtual machines with network transactions: Measurement and improvements," in *Proceedings of the 33rd IEEE International Conference on Computer Communications.* IEEE, 2014, pp. 1051–1059.

[61] J. Che, C. Shi, Y. Yu, and W. Lin, "A synthetical performance evaluation of openvz, xen and kvm," in *Proceedings of the IEEE Asia-Pacific Services Computing Conference.* IEEE, 2010, pp. 587–594.

[62] B. des Ligneris, “Virtualization of linux based computers: the linux-vserver project,” in *Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications.* IEEE, 2005, pp. 340–346.

[63] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, “Cost of virtual machine live migration in clouds: A performance evaluation,” in *Proceedings of the 1st International Conference on Cloud Computing.* Springer, 2009, pp. 254–265.

[64] D. Kapil, E. S. Pilli, and R. C. Joshi, “Live virtual machine migration techniques: Survey and research challenges,” in *Proceedings of the 3rd IEEE International Advance Computing Conference.* IEEE, 2013, pp. 963–969.

[65] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, “Remedy: Network-aware steady state VM management for data centers,” in *Proceedings of the 11th International IFIP TC 6 Networking Conference.* Springer, 2012, pp. 190–204.

[66] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, “Secondnet: a data center network virtualization architecture with bandwidth guarantees,” in *Proceedings of the 6th International conference on emerging Networking Experiments and Technologies.* ACM, 2010, pp. 15:1–15:12.

[67] W. Iqbal, M. N. Dailey, and D. Carrera, “Sla-driven dynamic resource management for multi-tier web applications in a cloud,” in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.* IEEE, 2010, pp. 832–837.

[68] C. Vecchiola, S. Pandey, and R. Buyya, “High-performance cloud computing: A view of scientific applications,” in *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks.* IEEE, 2009, pp. 4–16.

[69] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[70] C. Mastroianni, M. Meo, and G. Papuzzo, “Probabilistic consolidation of virtual machines in self-organizing cloud data centers,” *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 215–228, 2013.

[71] WordPress - Blog Tool, Publishing Platform, and CMS. [Online]. Available: https://wordpress.org/

[72] Microsoft Azure: Cloud Computing Platform & Services. [Online]. Available: http://www.microsoft.com/azure/

[73] Rackspace: Managed Dedicated & Cloud Computing Services. [Online]. Available: https://www.rackspace.com/

[74] G. Wang and T. E. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *Proceedings of the 29th IEEE International Conference on Computer Communications.* IEEE, 2010, pp. 1–9.

[75] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 460–471, 2010.

[76] L. Zhao, A. Liu, and J. Keung, "Evaluating cloud platform architecture with the care framework," in *Proceedings of the 17th Asia Pacific Software Engineering Conference.* IEEE, 2010, pp. 60–69.

[77] M. Klems, D. Bermbach, and R. Weinert, "A runtime quality measurement framework for cloud database service systems," in *Proceedings of the 8th International Conference on the Quality of Information and Communications Technology.* IEEE, 2012, pp. 38–46.

[78] D. Bermbach and S. Tai, "Eventual consistency: How soon is eventual? an evaluation of amazon s3's consistency behavior," in *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing.* ACM, 2011, p. 1.

[79] D. Bermbach, L. Zhao, and S. Sakr, "Towards comprehensive measurement of consistency guarantees for cloud-hosted data storage services," in *Proceedings of the 5th TPC Technology Conference on Performance Evaluation and Benchmarking.* Springer, 2013, pp. 32–47.

[80] C. A. Curino, D. E. Difallah, A. Pavlo, and P. Cudre-Mauroux, "Benchmarking OLTP/web databases in the cloud: The OLTP-bench framework," in *Proceedings of the 4th International workshop on Cloud data management.* ACM, 2012, pp. 17–20.

[81] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing.* ACM, 2010, pp. 143–154.

[82] A. Floratou, J. M. Patel, W. Lang, and A. Halverson, "When free is not really free: What does it cost to run a database workload in the cloud?" in *Proceedings of the 3rd TPC Technology conference on Topics in Performance Evaluation, Measurement and Characterization.* Springer, 2011, pp. 163–179.

[83] K. Huppler, "The art of building a good benchmark," in *Proceedings of the 1st TPC Technology Conference on Performance Evaluation and Benchmarking.* Springer, 2009, pp. 18–30.

[84] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the cloud: What it should, can, and cannot be," in *Proceedings of the 4th TPC Technology Conference on Performance Evaluation and Benchmarking.* Springer, 2012, pp. 173–188.

[85] D. Kossmann, T. Kraska, and S. Loesing, "An evaluation of alternative architectures for transaction processing in the cloud," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data.* ACM, 2010, pp. 579–590.

[86] D. Kossmann and T. Kraska, "Data management in the cloud: promises, state-of-the-art, and open questions," *Datenbank-Spektrum*, vol. 10, no. 3, pp. 121–129, 2010.

[87] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early observations on the performance of Windows Azure," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing.* ACM, 2010, pp. 367–376.

[88] T. Chen and R. Bahsoon, "Self-adaptive and sensitivity-aware qos modeling for the cloud," in *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* IEEE Press, 2013, pp. 43–52.

[89] W. Dawoud, I. Takouna, and C. Meinel, "Dynamic scalability and contention prediction in public infrastructure using internet application profiling," in *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science.* IEEE, 2012, pp. 208–216.

[90] H. Wu, A. N. Tantawi, and T. Yu, “A self-optimizing workload management solution for cloud applications,” in *Proceedings of the 20th IEEE International Conference on Web Services*. IEEE, 2013, pp. 483–490.

[91] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, “Adaptive resource provisioning for read intensive multi-tier applications in the cloud,” *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871–879, 2011.

[92] 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*, vol. 28, no. 5, pp. 755–768, 2012.

[93] B. Guenter, N. Jain, and C. Williams, “Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning,” in *Proceedings of the 30th IEEE International Conference on Computer Communications*, 2011, pp. 1332–1340.

[94] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization.* Athena Scientific Belmont, MA, 1997, vol. 6.

[95] A. J. Younge, G. Von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, “Efficient resource management for cloud computing environments,” in *Proceedings of the International Conference on Green Computing.* IEEE, 2010, pp. 357–364.

[96] E. Feller, C. Rohr, D. Margery, and C. Morin, “Energy management in iaas clouds: A holistic approach,” in *Proceedings of the 5th IEEE International Conference on Cloud Computing.* IEEE, 2012, pp. 204–212.

[97] E. Feller, L. Rilling, and C. Morin, “Snooze: A scalable and autonomic virtual machine management framework for private clouds,” in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* IEEE Computer Society, 2012, pp. 482–489.

[98] B. Kaur and A. Kaur, “An efficient approach for green cloud computing using genetic algorithm,” in *Proceedings of the 1st International Conference on Next Generation Computing Technologies.* IEEE, 2015, pp. 10–15.

[99] N. Kord and H. Haghighi, “An energy-efficient approach for virtual machine placement in cloud based data centers,” in *Proceedings of the 5th Conference on Information and Knowledge Technology.* IEEE, 2013, pp. 44–49.

[100] M. A. H. Monil, R. Qasim, and R. M. Rahman, "Energy-aware vm consolidation approach using combination of heuristics and migration control," in *Proceedings of the 9th International Conference on Digital Information Management*. IEEE, 2014, pp. 74–79.

[101] N. K. Sharma and R. M. R. Guddeti, "On demand virtual machine allocation and migration at cloud data center using hybrid of cat swarm optimization and genetic algorithm," in *Proceedings of the 5th International Conference on Eco-friendly Computing and Communication Systems*. IEEE, 2016, pp. 27–32.

[102] R. A. da Silva and N. L. da Fonseca, "Energy-aware migration of groups of virtual machines in distributed data centers," in *Proceedings of the 59th IEEE Global Communications Conference*. IEEE, 2016, pp. 1–6.

[103] W. Wang, Y. Jiang, and W. Wu, "Multiagent-based resource allocation for energy minimization in cloud computing systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 205–220, 2017.

[104] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *Proceedings of the 31st IEEE International Conference on Computer Communications*. IEEE, 2012, pp. 963–971.

[105] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2012, pp. 498–506.

[106] H. Chen, H. Kang, G. Jiang, and Y. Zhang, "Network-aware coordination of virtual machine migrations in enterprise data centers and clouds," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2013, pp. 888–891.

[107] K.-T. Chen, C. Chen, and P.-H. Wang, "Network aware load-balancing via parallel vm migration for data centers," in *Proceedings of the 23rd International Conference on Computer Communication and Networks*. IEEE, 2014, pp. 1–8.

[108] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[109] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling parallel applications in cloud simulations," in *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC).* IEEE, 2011, pp. 105–113.

[110] L. Cui, F. P. Tso, D. P. Pezaros, W. Jia, and W. Zhao, "Plan: Joint policy-and network-aware vm management for cloud data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1163–1175, 2017.

[111] Ns-3. [Online]. Available: http://www.nsnam.org

[112] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "VMFlow: Leveraging VM mobility to reduce network power costs in data centers," in *Proceedings of the 10th International IFIP TC 6 Networking Conference.* Springer, 2011, pp. 198–211.

[113] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Proceedings of the 9th International Conference on Grid and Cooperative Computing.* IEEE, 2010, pp. 87–92.

[114] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *Proceedings of the 2009 ICSE workshop on software engineering challenges of cloud computing.* IEEE Computer Society, 2009, pp. 9–14.

[115] H. Liu and B. He, "Vmbuddies: Coordinating live migration of multi-tier applications in cloud environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1192–1205, 2015.

[116] JMOB - RUBBoS Benchmark. [Online]. Available: http://jmob.ow2.org/rubbos.html

[117] H. T. Vu and S. Hwang, "A traffic and power-aware algorithm for virtual machine placement in cloud data center," *International Journal of Grid & Distributed Computing*, vol. 7, no. 1, pp. 350–355, 2014.

[118] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster computing*, vol. 16, no. 1, pp. 65–75, 2013.

[119] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," in *Proceedings of the 53rd IEEE Global Communications Conference.* IEEE, 2010, pp. 1–5.

[120] S. Alhiyari and A. El-Mousa, "A network and power aware framework for data centers using virtual machines re-allocation," in *Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies.* IEEE, 2015, pp. 1–6.

[121] 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.

[122] W.-C. Lin, C.-H. Liao, K.-T. Kuo, and C. H.-P. Wen, "Flow-and-vm migration for optimizing throughput and energy in sdn-based cloud datacenter," in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science*, vol. 1. IEEE, 2013, pp. 206–211.

[123] Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/

[124] The Network Simulator - ns - 2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[125] V. R. Reguri, S. Kogatam, and M. Moh, "Energy efficient traffic-aware virtual machine migration in green cloud data centers," in *Proceedings of the 2nd IEEE International Conference on High Performance and Smart Computing, IEEE International Conference on Intelligent Data and Security, and 2nd IEEE International Conference on Big Data Security on Cloud.* IEEE, 2016, pp. 268–273.

[126] J. Huang, K. Wu, and M. Moh, "Dynamic virtual machine migration algorithms using enhanced energy consumption model for green cloud data centers," in *Proceedings of the International Conference on High Performance Computing & Simulation.* IEEE, 2014, pp. 902–910.

[127] I. Takouna, R. Rojas-Cessa, K. Sachs, and C. Meinel, "Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers," in *Proceedings of the 6th IEEE/ACM International Conference on Utility and Cloud Computing.* IEEE Computer Society, 2013, pp. 251–255.

[128] I. Takouna, W. Dawoud, and C. Meinel, "Analysis and simulation of hpc applications in virtualized data centers," in *Proceedings of the IEEE International Conference on Green Computing and Communications.* IEEE, 2012, pp. 498–507.

[129] M. Harchol-Balter and A. B. Downey, “Exploiting process lifetime distributions for dynamic load balancing,” *ACM Transactions on Computer Systems*, vol. 15, no. 3, pp. 253–285, 1997.

[130] The 15 Best Blogging Platforms on the Web Today. [Online]. Available: http://thenextweb.com/apps/2013/08/16/best-blogging-services/

[131] L. Wolfe. Five key benefits of using wordpress for your website. [Online]. Available: https://www.thebalance.com/five-key-benefits-of-using-wordpress-for-your-website-3515362

[132] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009.

[133] K. Huppler, “Benchmarking with your head in the cloud,” in *Proceedings of the 3rd TPC Technology Conference on Topics in Performance Evaluation, Measurement and Characterization.* Springer-Verlag, 2011, pp. 97–110.

[134]

[135] J. Russell. The 15 Best Blogging Platforms on the Web Today. [Online]. Available: http://thenextweb.com/apps/2013/08/16/best-blogging-services/

[136] T. Holz, S. Marechal, and F. Raynal, “New threats and attacks on the world wide web,” *IEEE Security & Privacy*, vol. 4, no. 2, pp. 72–75, 2006.

[137] A. K. Singh. WordPress Use Cases. [Online]. Available: http://www.slideshare.net/teamphp/word-press-use-cases

[138] Selenium - Web Browser Automation. [Online]. Available: http://docs.seleniumhq.org/

[139] A. H. Borhani. WPressClient Source Code. [Online]. Available: http://figshare.com/articles/WPressClient_source_code/978486

[140] D. T. McWherter, B. Schroeder, A. Ailamaki, and M. Harchol-Balter, “Priority mechanisms for OLTP and transactional web applications,” in *Proceedings of the 20th International Conference on Data Engineering*, 2004, pp. 535–546.

[141] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in dynamic structured p2p systems," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4. IEEE, 2004, pp. 2253–2262.

[142] K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voss, "Introducing risk management into the grid," in *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing.* IEEE, 2006, pp. 28–28.

[143] H. Rong, H. Zhang, S. Xiao, C. Li, and C. Hu, "Optimizing energy consumption for data centers," *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 674–691, 2016.

[144] D. Huang, B. He, and C. Miao, "A survey of resource management in multi-tier web applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1574–1590, 2014.

[145] UMass trace repository. [Online]. Available: http://traces.cs.umass.edu/index.php/

[146] B. Godfrey. Repository of availability traces, 2010. [Online]. Available: http://pbg.cs.illinois.edu/availability/

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

[148] Standard performance evaluation corporation. [Online]. Available: https://www.spec.org/power_ssj2008/

[149] L. Rongheng, W. Budan, Y. Fangchun, Z. Yao, and H. Jinxuan, "An efficient adaptive failure detection mechanism for cloud platform based on volterra series," *China Communications*, vol. 11, no. 4, pp. 1–12, 2014.

[150] C. Engelmann, G. R. Vallee, T. Naughton, and S. L. Scott, "Proactive fault tolerance using preemptive migration," in *Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing.* IEEE, 2009, pp. 252–257.

[151] J. Lee, H. Yu, and J. Gil, “A virtual machine migration management for multiple datacenters-based cloud environments,” in *Advanced Computer Science and Information Technology.* Springer, 2011, pp. 198–205.

[152] T. S. Kang, M. Tsugawa, J. Fortes, and T. Hirofuchi, “Reducing the migration times of multiple vms on wans using a feedback controller,” in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum.* IEEE, 2013, pp. 1480–1489.

[153] L. Nie, C. Xie, Y. Yin, and X. Li, “Context-aware qos assurance for smart grid big data processing with elastic cloud resource reconfiguration,” in *Proceedings of the ICA3PP International Workshops and Symposiums on Algorithms and Architectures for Parallel Processing.* Springer, 2015, pp. 177–186.

[154] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, “Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 732–749, 2011.

[155] B. Bacci, M. Danelutto, S. Pelagatti, and M. Vanneschi, “Skie: a heterogeneous environment for hpc applications,” *Parallel Computing*, vol. 25, no. 13, pp. 1827–1852, 1999.

[156] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, “Case study for running hpc applications in public clouds,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing.* ACM, 2010, pp. 395–401.

[157] J.-K. Yu, M. La Rota, R. Kantety, and M. Sorrells, “Est derived ssr markers for comparative mapping in wheat and rice,” *Molecular Genetics and Genomics*, vol. 271, no. 6, pp. 742–751, 2004.

[158] A. Zlot, A. Newell, K. Silvey, and K. Arail, “Peer reviewed: Addressing the obesity epidemic: A genomics perspective,” *Preventing chronic disease*, vol. 4, no. 2, 2007.

[159] A. Jaiswal and A. Upadhyay, “An enhanced framework of genomics using big data computing,” in *Computer, Communication and Control (IC4), 2015 International Conference on.* IEEE, 2015, pp. 1–7.

[160] P. C. Church and A. M. Goscinski, “A survey of cloud-based service computing solutions for mammalian genomics,” *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 726–740, 2014.