2019

Energy-efficient resources utilization algorithm in cloud data center servers.

Kenga, Derdus Mosoti
Faculty of Information Technology
Strathmore University

Energy-Efficient Resources Utilization Algorithm in Cloud Data Center Servers

Kenga Mosoti Derdus

Submitted in total fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Technology at Strathmore University

Faculty of Information Technology
Strathmore University
Nairobi, Kenya

September, 2019

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Kenga Mosoti Derdus
…….....
September 2019

Approval

The thesis of Kenga Mosoti Derdus has been approved for examination by:

Dr. Vincent O. Omwenga,
Faculty of Information Technology,
Strathmore University

Prof. Patrick J. Ogao,
Faculty of Engineering Science and Technology,
Technical University of Kenya

Abstract

In recent years, the use of cloud computing has increased exponentially to satisfy computing needs and this is attributable to its success in delivering service on a pay-as-you-go basis. As a result, Cloud Service Providers (CSPs) are putting up more data centers to meet the demand. However, the high amount of energy consumed by cloud datacenter servers has raised concern because CSPs experience high operating costs (electricity bills), which reduces profits. The cause of high energy usage in datacenter servers is energy wastage, which results from low level of server utilization. This problem is currently addressed through Virtual Machine (VM) consolidation and Dynamic Voltage and Frequency Scaling (DVFS). Unfortunately, VM consolidation does consider workload types and VM sizes, which are factors that affect level of server utilization. On the other hand, DVFS is designed for processor bound tasks because dynamic power ranges for other computing resources such as memory are narrower.

In this study, the effect of workload types (heterogeneous or homogeneous) running in VM and VM sizing on datacenter server energy consumption were investigated. The results obtained from conducted experiments show that heterogeneous workload is consolidation friendly as compared to homogeneous workloads from a datacenter energy consumption perspective. Further, a review of literature discovered that oversized VMs lead to the low level of server utilization and thus leads to energy wastage. Consequently, VM allocation and VM sizing algorithms have been proposed and tested. The VM allocation algorithm co-locates heterogeneous workload whereas VM sizing algorithm is used for VM right-sizing. In order test the applicability of the proposed algorithms in the cloud, the algorithms were evaluated using simulations on a cloud simulator (CloudSim Plus) using workloads logs obtained from a production datacenter (Grid Workload Archive Trace 13 (GWA-T-13)).

Results on the evaluations carried on the designed VM allocation algorithm showed that datacenter server energy consumption reduced by 4%, 11%, 17% when compared with Worst Fit (WF), First Fist (FF) and Best Fit (BF) VM allocation algorithms respectively. On the other hand, the VM sizing algorithm reduced energy consumption, memory wastage, and CPU wastage by at least 40%, 61% and 41% respectively. From the results, we concluded that workload types and VM sizes affect level of server utilization, which in turn determines the amount of energy consumption. Thus, the right workload types combined with right VM sizing leads to high level of server utilization leading to energy savings.

Acknowledgement

A PhD study is a long journey, which, without the support from other people, can be difficult to attain. In this regard, I would like to thank the following people:

1) My PhD supervisors, Dr. Vincent Omwenga and Prof. Patrick Ogao for their tireless support during the entire period of study. The support of these two gentlemen is invaluable and was beyond academic – when I almost gave up, you were there to encourage me. When I was stuck in understanding certain concepts, you were always available to explain and point to the easiest ways of understanding new ideas. God bless you and your families.
2) DAAD organization for offering me the financial and training support to pursue my PhD degree.
3) The entire family of Faculty of Information Technology headed by Dr. Joseph Orero for your moral support and encouragement during this journey. I especially thank the ICS thematic group for the advice you gave to me every Thursday at our meetings. Special thanks go to my colleagues Esther, Allan, Henry, Shabaya, Machanje, Danny and Nicodemus, who were on the same journey as me, for the endless moral support you offered. I wish you well as you finish your studies too.
4) Entire Strathmore fraternity led by Prof. Odhiambo and especially the school of graduate studies for providing a friendly environment for PhD studies.
5) My friends Antony, Khajira, Hezzy and Okara for your constructive comments and encouragement.
6) My entire family particularly my wife Mercy, my mother Naom, my brother Lewis and my sisters Winnie and Prisca for your prayers and infinite love.

Dedication

To my loving wife Mercy and daughter Harmony

Table of Contents

List of Figures

List of Table

Abbreviations and Acronyms

ANN – Artificial Neural Network

API – Application Programming Interface

ARIMA - Autoregressive Integrated Moving Average

AWS – Amazon Web Service

BF – Best Fit

CMS – Cloud Management Software

CoV – Coefficient of Variance

CPU – Central Processing Unit

CSP – Cloud Service Provider

DVFS - Dynamic Frequency and Voltage Scaling

EBS - Elastic Block Store

EC2 – Elastic Compute Cloud

FF – First Fit

GCE - Google Compute Engine

GCT – Google Cluster Trace

GPU - Graphics Processing Unit

GWA - Grid Workload Archive

GWA-T - Grid Workload Archive Trace

GWA-T-13 - Grid Workload Archive Trace 13

I/O – Input Output

IaaS – Infrastructure as a Service

ICT – Information Communication Technology

IT – Information Technology

KVM - Kernel-based Virtual Machine

NF – Next Fit

NRDC - Natural Resources Defense Council

PaaS – Platform as a Service

PC – Personal Computer

PDU – Power Distribution Unit

PM – Physical Machine

PUE - Power Usage Effectiveness

QoS – Quality of Service

REST - Representational State Transfer

ROI – Return on Investment

SaaS – Software as a Service

SIGAR - System Information Gatherer and Reporter

SLA – Service Level Agreement

SMA – Simple Moving Average

SPECPower - Standard Performance Evaluation Corporation Power

SR – Success Rate

SSD – Solid State Disk

TCO – Total Cost of Ownership

VM – Virtual Machine

VMM – Virtual Machine Manager

WEKA - Waikato Environment for Knowledge Analysis

WF – Worst Fit

Definition of Terms

Cloud computing – a model or concept 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. The resources are accessed over the internet (NIST, 2011).

Cloud environment - an IT service provision entity that is powered by cloud computing. A cloud environment consists of physical machines (servers) in a data center, networks, network devices, cooling infrastructure and software (Tesfatsion, 2018).

Computing resource – is a physical or virtual component of limited availability processor, memory, network connection and input and output devices (Tesfatsion, 2018).

datacenter – is a facility that houses computing equipment such as servers, switches, routers, firewalls and other supporting components such as cooling facility, fire suppression facility and backup equipment (Tesfatsion, 2018).

Dynamic Voltage and Frequency Scaling (DVFS) - DVFS is an energy saving technique in computer architecture that is used to save energy when the server load is low (Sareh, 2016). In this technique, the frequency and voltage of the CPU are scaled dynamically to match the amount of server load. This technique is used to achieve computing proportionality.

Homogeneous workload – a specific amount of application workload, which tends to consume more of the same computing resources (Dhiman, 2011). If the application workload consistently uses the processor and not any other resources, it is termed as homogeneous workload and can be labeled as processor bound workload.

Heterogeneous workload – a specific amount of application workload, which tends to consume different computing resources (Dhiman, 2011). If the application

workload uses all the resources in a fairly balanced manner, it is termed as heterogeneous workload.

Physical machine - is the physical hardware-based device and it holds the physical compute resources. This term is used to differentiate the hardware-based computer from software-based machines (virtual machines) (Sharma, 2014).

Proportional computing – is a situation where energy consumed by datacenter servers is proportional to server load. This is the opposite of un-proportional computing, where energy consumed by the datacenter server is not proportional to server load – energy consumption is higher even at low server load (Sareh, 2016).

Virtual machine – is an emulation of the physical machine using a software (virtual machine monitor) and it uses virtual resources made available by the virtual machine monitor (Sharma, 2014).

Virtual Machine consolidation – this is the process of ensuring that virtual machines use the least amount of servers while maintaining the required level of performance (Anton, 2013). This process involves virtual machine allocation, which the process of determining the server to host incoming virtual machines and virtual machine migration, which is the process of moving a virtual machine from one server to another for various reasons such as load balancing and energy savings.

Virtual machine monitor - is a software layer, which induces the partitioning capability in physical machines to give rise to virtual machines (Sharma, 2014).

Virtual Machine (VM) sizing – is the process of determining the sizes of computing resources (processor, memory, hard disk and network bandwidth) to be allocated to any given virtual machines (Sareh, 2016). A VM size is the measure of resources (processor, memory, hard disk and network bandwidth) allocated to a given VM. And oversized VM is that which whose resource demand is much lower than the resources allocated.

Workload benchmark – it is a software package with a set of characteristics that are used to test the performance of an object such as a computer (Smith & Sommerville, 2011).

Workload – is the amount of processing that a computer has been given to perform. A workload consists of application demanding specific computer resources such as processor, memory and network bandwidth (Ismael et al., 2013)

Workload type – the behaviors of workload in terms of resources usage i.e. if a workload is homogeneous or heterogeneous.

Workload characterization - is the process of understanding the characteristics of a given application workload such as resource consumption patterns and task failure rates by analyzing application usage logs (Shen, Beek, & Iosup, 2015). Homogeneous workloads and heterogeneous workloads are two types of workloads. From an energy consumption perspective, a consolidation friendly workload type is that which does not lead to excessive data center energy consumption.

Chapter 1

Introduction

1.1 Background to the study

Cloud computing has gained a lot of interest from both small and big, academic and commercial organizations because of its success in delivering service on a pay-as-you-go basis. To address this need, cloud service providers (CSPs) are hosting many servers in public cloud datacenters to provide the levels of computing power that is demanded from customers (Chaima, 2014). Additionally, organizations are putting up private cloud data centers to be able to control their own computing needs (Chen, et al., 2015). Unfortunately, the amount of energy consumed by the datacenters is a worrying concern to cloud computing practitioners. According to Dayarathna & Wen, (2016), an average datacenter (covering around 100,000 sq ft.) consumes as much energy as 25, 000 households in the United States (US). As a result, energy consumption by data centers account for 3% of globally electrical energy consumption (Rallo, 2014) and is estimated to triple by the year 2020 (Salam, Karim, & Ali, 2018). The excessive energy consumption has drastically increased the Total Cost of Ownership of cloud infrastructure resulting in low Return on Investment (RoI). According to Albert et al. (2009), power bills dominate the operating costs in a datacenter. For example, the cost of energy in Amazon datacenters has reached 42% of its operating costs (Salam, Karim, & Ali, 2018). *Figure 1.1* shows capital expenditure on servers and power and cooling infrastructure, operating costs such as power bills and other operating costs. Power bill is the highest consumer of operating costs.

Apart from low RoI, excessive energy consumption has a negative impact on the environment, which is carbon dioxide (CO_2) emission. According to Anton (2013), the ICT industry is estimated to contribute about 2% of global CO_2

emission, which contributes greatly to the greenhouse effect. In fact, this amount of emission is equivalent to the aviation industry.

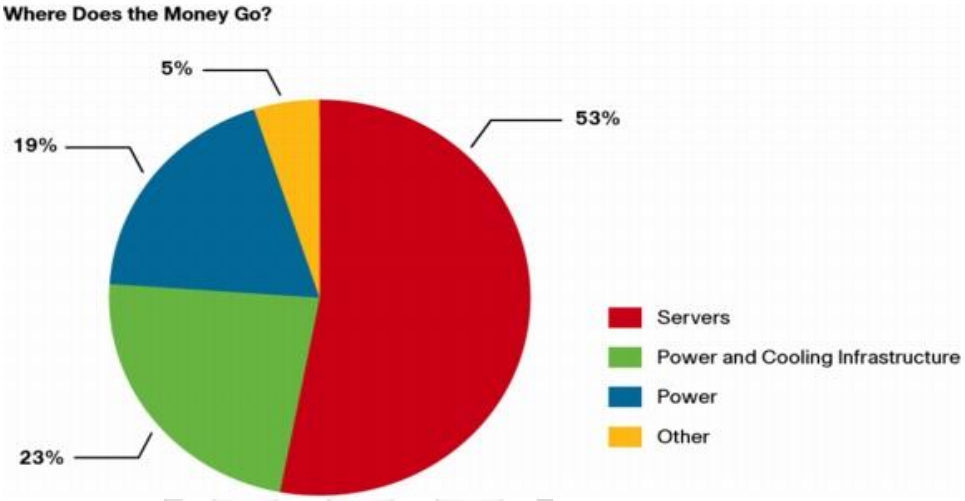


Figure 1.1: Where does the money go in a datacenter? (Adopted fromCisco, 2013)

As illustrated in Figure 1.2, worldwide energy consumption by data centers has risen steadily from the year 2000 to 2010. In 2010, datacenter accounted for about 1.5% of total energy consumed worldwide (Anton & Rajkumar, 2010). *Figure 1.3*, which compares projections of electricity usage by data centers in the US and globally, shows that consumption is set to increase drastically towards the year 2020. It is also estimated that by the year 2020, the US alone will release about 150 million tonnes of CO_2 as a result of its datacenter consuming electricity (Khosravi, 2017). In another source, as seen in Figure 1.4, the energy consumption in data centers is projected to increase as other sources of energy consumption decreases such as television and personal computers (PCs).

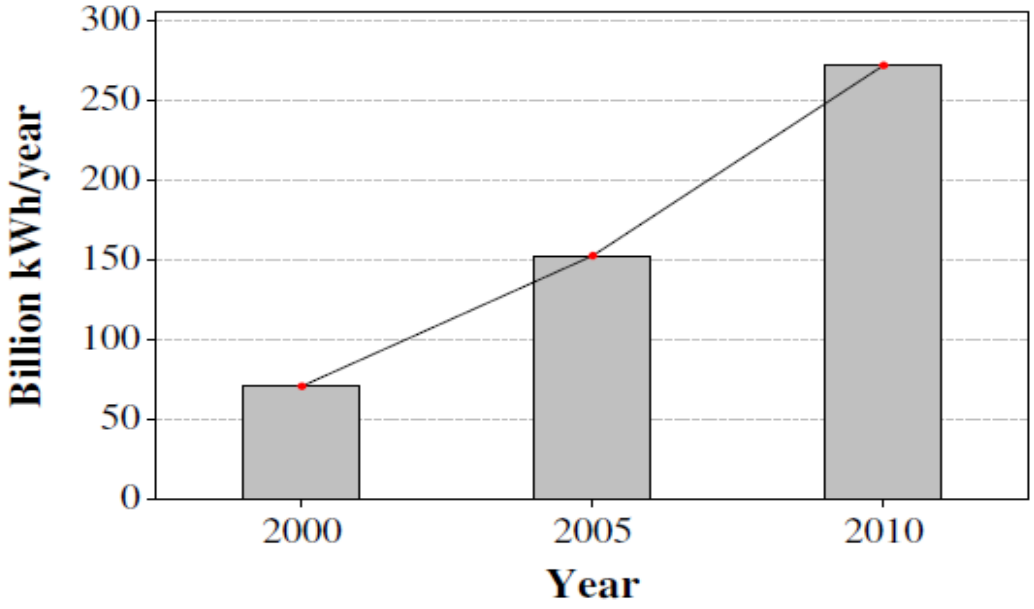


Figure 1.2: The worldwide data center energy consumption 2000-2010 (Adopted from Anton, 2013)

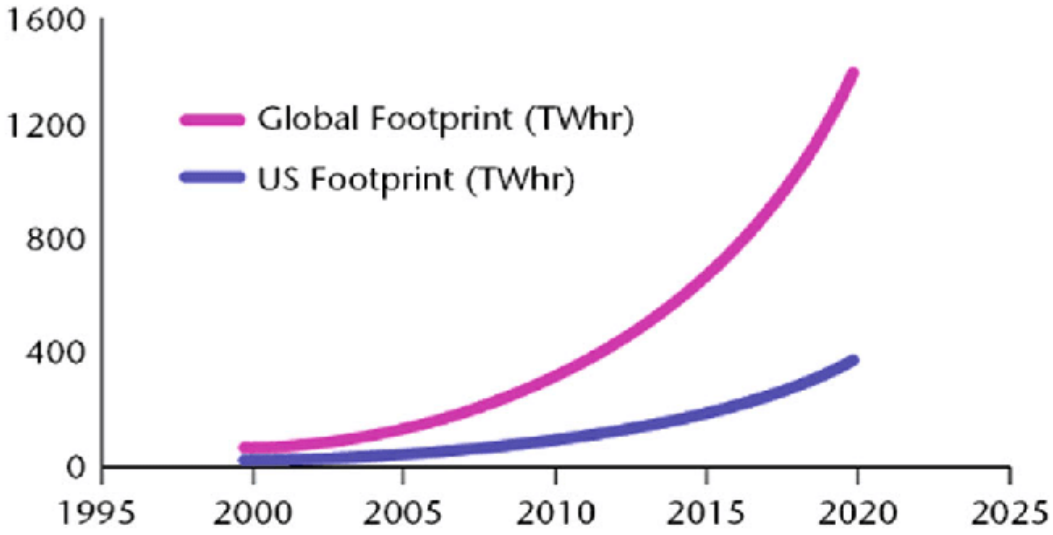


Figure 1.3: Projection of datacenters electricity use (Adopted from Salam, Karim, & Ali, 2018)

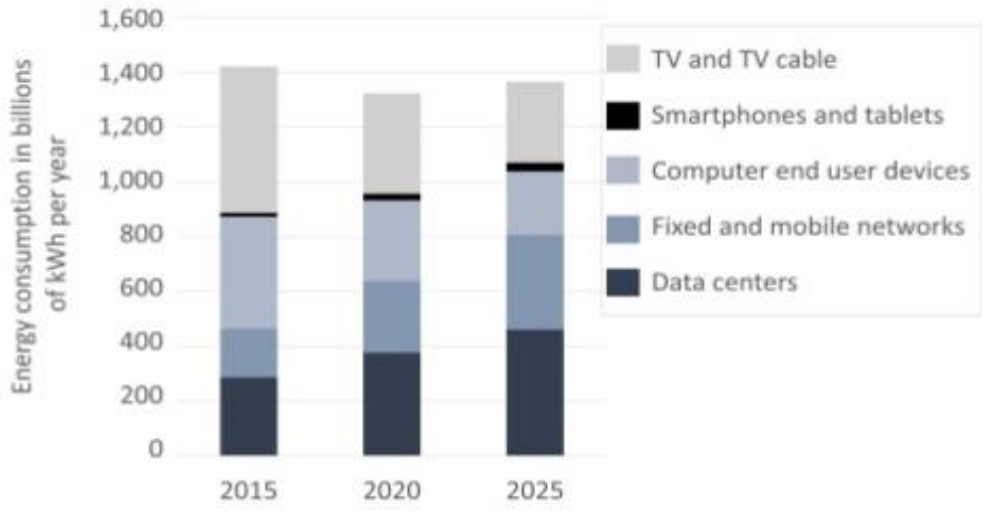


Figure 1.4: Global Information Technology (IT) energy consumption in 2015 and forecast for 2020 and 2025 in various industries (Adopted from Hintemann & Clausen, 2016)

According to Ullah, Hassan and Khan, (2017), at least 56% of enterprises used the cloud by 2014. It is also estimated that 83% of all computing workloads will be processed in the cloud by 2020 (Columbus, 2018). Similarly, according to Bezos' law, in the history of cloud computing, it is observed that the cost of computing power reduces by 50% every 3 years (O'Connor, 2014). Because of the reduced cost of executing workloads in the cloud, it is expected that businesses will abandon their data centers and move to the public cloud as a way of saving money. Equally, new businesses currently using traditional computing will adopt cloud computing. This traffic to the public cloud means that energy consumption in the cloud will continue to be a concern.

Consumption of excessive energy is not really a problem, but how efficient the energy is being used. For many decades, the primary concern of computing infrastructure designers has been the performance in terms of CPU speed (Anton, 2013). Because of this, the amount of power drawn by computing device has gone up especially in commercial computing devices such as datacenters. The main causes of high power consumption are low server utilization, idle power wastage

and small firms not adopting energy efficient solutions (Sareh, 2016). The fact that most data centers do not utilize even 50% of their computing capability leads to wastage of idle power. This is true especially because an idle server can consume over 70% of their peak energy (Chaima, 2015; Anton, 2013). This is a case of resource underutilization. The case where lowly utilized servers consuming more energy is related to energy un-proportionality, which means that when server load is low, energy consumption is still high. Proportional computing is achieved by Dynamic Frequency and Voltage Scaling (DVFS). DVFS is an energy saving technique in computer architecture that is used to save energy when the server load is low (Barroso and Holzle, 2007). In this technique, the frequency and voltage of the CPU are scaled dynamically to relate with the amount of server load. DVFS is hardware-based technique and works well only on CPU bound tasks because dynamic power ranges for other components such as memory are narrower (Anton, 2013). Wastage of idle energy is also caused when workloads executed in a server dominates one resource and leaves other computing resources idle (Dhiman, Mihic, & Rosing, 2010).

Another way which improves resource utilization is dynamic consolidation of Virtual Machines (VMs) (Anton, 2013). This is possible by using virtualization technology, which enables CSPs to run many VM instances in the same server thus improving resource utilization. When resource utilization is high, idle energy wastage is reduced thus improving efficiency. However, virtualization alone cannot achieve higher resource utilization if applications running in the VM are using fewer resources than those allocated to the VM.

To address the problem of energy wastage as a result of low server utilization, which is the focus of this study, solutions need to be designed, which ensures that server utilization is averagely high to save idle energy. Such solutions need to consider the concept of virtualization, which is the main technology in the cloud as well as how resources are allocated to application workloads in the cloud.

Maintaining an acceptable high server utilization means that the solutions designed need to be aware of the number of resources required by cloud workloads at all times. Due to cloud dynamicity, such solutions need to understand the nature of cloud workloads and forecast their future resource requirements for purposes of in-time resource provisioning (Ullah, Hassan, & Khan, 2017). According to Delimitrou (2015), cloud dynamicity is called *performance unpredictability*, which is as a result of platform heterogeneity as machines are replaced over the lifetime of a datacenter, resource contention as a result of applications sharing the underlying hardware and unpredictable fluctuations of user load. On the other hand, resource underutilization can occur where users have the responsibility to reserve resources required for their applications running in VMs. Unfortunately, for inexperienced users, more resources, than required, are assigned to VMs leading to server underutilization (Patel, et al., 2015). This is called *resource overprovisioning*.

Although the designed solutions need to scale to cover the different cloud service models, it is prudent to adopt it to a particular one due to the competing interests in the different dimensions that make solution design complex (Chaima, 2014). The service models include Infrastructures as a service (IaaS), Platform as a Service (PaaS) and Software as a service (SaaS). In this study, a solution tailored to IaaS but which can be modified to fit into other service models is proposed.

The motivation to carry out this research is based on the understanding that the classic techniques of addressing the aforementioned challenges are getting overwhelmed as the cloud computing architectures get complex. Moreover, certain techniques and technologies such as automatic hypervisor resource provisioning, application of artificial neural network for predicting datacenter resource usage and datacenter simulation software, are maturing and can be helpful. For instance, a Xen hypervisor can provision resources to an IaaS VM automatically without human intervention using CPU hot-plug and dynamic

memory (Yazdanov & Fetzer, 2012). For resource planning and provisioning, artificial neural networks can predict a non-linear time series, which is a common phenomenon with datacenter resource usage. Testing cloud computing solutions are now cheaper owing to the development of software, such as CloudSim, which simulates the cloud (Rodrigo et al., 2011). The aforementioned developments create a favorable environment for successful research in the area of cloud computing.

1.2 Problem statement

The adoption of cloud computing is becoming indispensable in the modern IT world. It is also definite that many organizations will abandon their traditional in-house IT infrastructure and use the cloud because prices of cloud computing units are dropping. The CSPs have responded by putting up more data centers to ensure Quality of Service (QoS) and obeying Service Level Agreements (SLA). Unfortunately, cloud data centers consume a lot of energy and are responsible for consuming about 3% of global electrical energy consumption (Rallo, 2014). As a result, CSPs experience high operating costs and increased Total Cost of Ownership (TCO). For instance, power bills dominate the operating costs in a data center leading to low profits (Albert et al., 2009). In addition, energy consumed by datacenters contributes to about 2% of CO_2 emission to the environment (Anton, 2013). Energy consumed by data centers has also caused the cost of using cloud computing resources to be on the rise, which has a direct financial burden on cloud users.

The cause of energy wastage in data centers is a low level of server utilization, which leads to wastage of idle energy. The fact that servers are energy un-proportional compounds this problem. On the other hand, the low level of server utilization means that resources are wasted because more resources are provisioned than required. Moreover, energy wastage occurs because of the workload profiles co-located in a server. For instance, co-locating homogeneous

workloads causes resource contention as only one resource is dominated leading to loss of performance and wastage of idle power of other unutilized resources (Dhiman, 2011; Mohsen at al., 2011; Chen, et al., 2015; Sareh, 2016). Thus workload consolidation need to consider workload profiles. For IaaS cloud model, which is the focus of this study, there are two reasons for resource underutilization, which are 1) *Performance unpredictability* by cloud workloads and 2) *Resource overprovisioning* by inexperienced users (Chen, et al., 2015). *Performance unpredictability* requires that future resource usage by workloads be forecasted for provisioning to ensure that the exact resources required are provisioned. On the other hand, *resource overprovisioning* by inexperienced users requires that cloud platform analyzes resource usage by user workloads over time to be able to propose new resource allocation policies. From this point of view, the required solution must ensure that server resource utilization is at high levels and that co-located workloads profiles are friendly in terms of energy consumption.

The current techniques for addressing the problem of resource underutilization for energy savings are VM consolidation and DVFS. VM consolidation fails because if consolidation is performed without considering energy and performance profiles of application workloads, no energy savings are likely to be achieved (Dhiman, 2011). In addition, current approaches concentrate on maximizing host-level resource utilization to benefit CSP and have forgotten maximizing VM-level resources utilization to benefit both the CSP and cloud user. On the other hand, DVFS works well only on CPU bound tasks because dynamic power ranges for other components such as memory are narrower. Hence, there is a need to create new energy efficiency techniques, which considers application workload profiles for VM consolidation and maximizing server resource utilization at VM-level. Consolidating energy friendly VMs profiles ensures that idle energy is efficiently used. On the other hand, maximizing server resource

utilization at VM-level ensures that fewer physical servers are used to execute workloads – fewer physical servers will consume less energy.

1.3 General objective

The main objective of this research is to design an algorithm to reduce energy wastage in datacenters servers while processing application workloads.

1.4 Specific objectives

(i). To analyze how server related factors influence energy consumption in a cloud datacenter

(ii). To review techniques used to reduce energy consumption by cloud data center servers

(iii). To design and develop algorithms, which considers contextual server factors, to reduce energy consumption and wastage in cloud datacenter servers

(iv). To evaluate the performance of the proposed algorithm

1.5 Research questions

(i). How do server related factors influence energy consumption in a cloud datacenter?

(ii). What techniques are used to reduce energy consumption in a cloud datacenter?

(iii). How can an algorithm, which considers contextual server factors, to reduce energy consumption and wastage in cloud datacenter servers be designed and developed?

(iv). How can the performance of the proposed algorithm be evaluated?

1.6 Significance of the study

While cloud computing is indispensable in the modern IT world, its continued adoption will continue to increase the amount of energy consumption. This raises an array of questions; 1) how much CO_2 is released to the environment as a results of the energy consumed?, 2) how much money is used in power bills by CSPs and what is the implication of this on CSPs' profits and cost of cloud

computing resources passed to cloud users?, 3) what is the role of academic research in addressing the problem of energy consumption in a cloud computing environment? The proposed solution in this research is to reduce energy consumption and wastage in cloud datacenter servers by improving resource utilization. Thus, the contribution of this research is threefold: social-economic, policy and academic contributions.

1.6.1 Social-economics contribution

The successful design of algorithms, which reduce energy consumption and wastage, will lead to a reduction in power bills thus increasing CSPs' profits. This also reduces the cost of renting cloud computing resources from the side of cloud users. In addition, because all the energy used in data centers is not fossil-free, a reduction in energy consumption and wastage leads to a reduction in CO_2 emission (Data-Economy, 2019; Google, 2019). CO_2 is a greenhouses gas, which causes the greenhouse effect.

1.6.2 Policy contribution

From a policy perspective, the outcome of this research has an implication on global standards on data center energy efficiency metrics. For a long time, the two datacenter power efficiency metrics in use have been Power Usage Effectiveness (PUE) and data Center Efficiency (DCE) (The Green Grid, 2007). PUE is defined as the total power consumed by the data center divided by the power used by the IT equipment and DCE is the ratio of IT data center energy efficiency and is defined as the reciprocal of PUE. However, these two metrics do not show how well the energy drawn is used to process the IT load. In this research, we have shown that adopting Performance per Watt (PPW) as a data center power efficiency is possible and is the way to go. The advantage of PPW is that it shows the amount of IT load that is processed per watt of power drawn. Therefore, it is time for organizations such as The Green Grid to publish this metric to be adopted by CSPs. Moreover, this research has shown that it is possible to reduce the current

energy wastage in data centers. Therefore, it makes sense to increase carbon taxes for CSPs whose data centers are powered by electricity generated from coal.

1.6.3 Academic contribution

The outcome of this research will contribute immensely to the body of existing knowledge. This research provides alternative approaches for solving certain problems. For instance, while other researchers concentrate on maximizing usage of host-level resources as a way of saving energy (Anton, 2013), this research maximizing usage of VM-level resources. Based on the objectives that were set earlier, the main contributions of this study are summarized as follows:

1. A survey on; -
    a. Techniques used to reduce energy consumption and wastage in cloud datacenter servers.
    b. Statistical techniques for characterizing cloud workloads.
    c. The server related factors, which influence energy consumption and wastage in cloud data center servers.
2. An experimental approach for determining performance and power profiles of different types of cloud workloads. Phoromatic Test Suite (PTS) workload generator has been used and it has been discovered that heterogeneous workloads are more consolidation friendly that homogeneous workloads from an energy saving perspective.
3. A VM allocation algorithm that considers VM workload type power and performance profiles before consolidation. The designed VM allocations algorithm reduces energy wastage by data center servers by ensuring that heterogeneous workloads are co-located, which saves energy. The algorithm has been implemented in CloudSim Plus simulator for evaluation.
4. A VM sizing algorithm in multi-tenant IaaS public cloud. This algorithm avoids resource wastage and ensures that application workloads are

consolidated in the least possible number of physical servers thus saving on energy, which would have been consumed by extra servers.

1.7 Scope and limitations of the study

This study investigated efficient resource utilization in virtualized cloud environments by ensuring that server resources are not underutilized with the aim of reducing data center server energy consumption caused by executing cloud workloads. Efficient allocation of computing resources means that cloud workloads can be executed by using fewer physical servers, which leads to energy savings. Moreover, the study investigated co-location friendly (from energy savings perspective) workload types. The target cloud service model and deployment model in this study were IaaS and public multi-tenant respectively. IaaS cloud service model has been chosen because it is the most promising model for providing cloud services to small businesses and new entrants in the cloud space (Neha & Rishabh, 2015). Moreover, public multi-tenant deployment model was of interest because, according to Bezos' law, it is expected that businesses will abandon their private data centers and move to the public cloud as a way of saving money through sharing resources.

In IaaS, cloud users determine their application resource demands (VM size), which often leads to *resource overprovisioning*. Furthermore, the problem of *performance unpredictability* becomes compounded with user interactive applications such as search, which causes spikes. For this reason, this study used real workload resource usage traces for analysis to understand VM workload resource usage behavior for purposes of VM sizing and future resource usage prediction. This study used publicly available workload traces such as Grid Workload Archive Trace 13 (GWA-T-13) Materna because it was not possible to build large scale cloud for collecting cloud traces (Delf University, 2018). This study also used workload generators PTS (Phoronix Test Suite, 2018). Unlike other approaches, such as that in Anton (2013), which concentrated on maximizing host-

level resource utilization to benefit CSP, our approach concentrated in maximizing VM-level resource utilization to benefit both the CSP and cloud user.

Lastly, the proposed algorithms were evaluated using a cloud simulator called CloudSim Plus (Manoel et al., 2017). A cloud simulator was used because deploying a real cloud data center for testing purposes could have been costly. Besides, using a simulator allowed us to create a data center of any size and to change configurations at will with no cost. *Table 1.1* summarises the scope of this study.

Table 1.1: A summary of the study Scope and delimitation

Characteristic	**Scope**
Goal	Minimize energy usage through efficient resource utilization. The focus of energy savings is on data center servers.
Target service and cloud model	IaaS, public multi-tenant cloud
Energy saving technique	VM allocation/placement, VM sizing, VM resource usage prediction
Workload sources	Grid Workload Archive Trace 13 (GWA-T-13) Materna, workload generation using PTS
Workload analysis techniques	Use of statistical techniques such as clustering, correlations, percentiles and peak detection.
Resource usage prediction techniques	Time series analysis and forecasting using artificial neural networks (ANN)
Evaluation Technique	Using simulations - CloudSim Plus for VM allocation and sizing, Waikato Environment for Knowledge Analysis (WEKA) for ANN model evaluation (Waikato University , 2018).

1.8 Research methodology

The research philosophy in this study identifies that a scientific way is the ultimate way of establishing truth and objective reality. This reality is proven by data and relatively constant across time and can be tested empirically. Therefore, the research design adopted in this study is experimental. Prior to conducting experiments, a literature review was used to broaden the researcher's knowledge of the existing body of knowledge in the subject area and to identify the applicable methodology in this study. The methodology applied in this study can be categorized into four phases. Each phase has several consecutive steps as shown in Figure 1.5.

Phase 1 involved looking at cloud computing technology and how energy is utilized in cloud data centers and data center servers. In addition, experiments were conducted to understand power and performance profiles of different types of workloads. In phase 2, the insights from phase 1 were used to design the VM allocation algorithm, VM sizing algorithm and VM resource usage prediction algorithm. Finally, in phase 3, the algorithms designed in phase 2 and evaluated. VM allocation algorithm, VM sizing algorithm and VM resource usage prediction algorithm were evaluated using simulations. Additionally, conclusions and recommendations for the study were given in this phase.

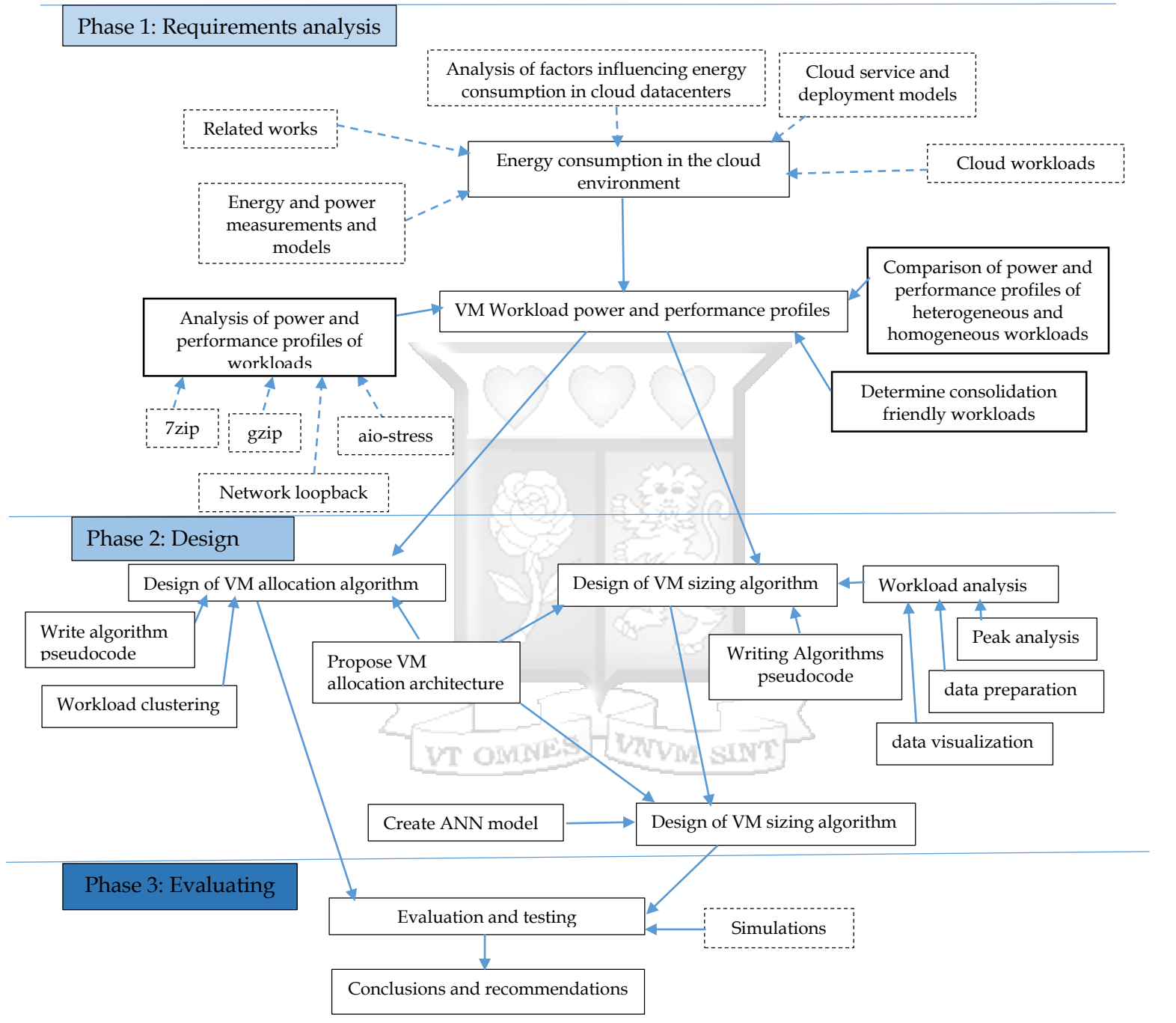


Figure 1.5: A summary of research steps

**The solid lines in Figure 1.5 represents tasks that were performed whereas dash-line boxes represent concepts, theories and other expert knowledge that we used in the study.*

1.9 Thesis organization

The remaining part of this thesis is divided into 6 chapters. A summary of the thesis structure is as shown in Figure 1.6. Chapter 2 and Chapter 3 are mainly concerned with the relevant literature relating to the subject of study. Specifically, Chapter 2 focusses on cloud computing technology such as deployment and service models as well as its supporting technologies such as virtualization. It also covers how energy is consumed in a cloud environment, the factors that affect energy consumption in the cloud and energy measurement and modelling in computing environments. The content of this chapter addresses the first objective of this thesis.

Chapter 3 of this thesis reviews the existing general techniques for energy efficiency in the cloud especially those that ensure efficient server resource utilization. This chapter also covers understanding cloud workloads, their sources and how their understanding can be used to achieve effective workload consolidation with the aim of reducing energy consumption. This chapter also touches on techniques such as artificial neural network (ANN) and Autoregressive Integrated Moving Average (ARIMA) and how they have been used to predict datacenter resource usage. Finally, this chapter touches on tools that can be used to simulate and test results in the space of cloud computing and datacenter resources usage prediction. Examples of the tools include a cloud simulator known as CloudSim Plus and Waikato Environment for Knowledge Analysis (WEKA). The content in this chapter addresses the second objective of this thesis.

In chapter 4, the experiment, which was designed to determine the effect of workload types on power consumption is described. This chapter also describes the experiment, which was used to determine the effect of level of consolidation on data center server power consumption and performance in workload processing. The implications of the results obtained from the experiments on VM machine allocation is also presented.

In chapter 5, the experimental results obtained in Chapter 4 are used to propose a VM allocation algorithm. This chapter also explains how the proposed VM allocation algorithm was designed, implemented and tested.

Further, in chapter 6, an architecture for virtual machine (VM) sizing, which targets multi-tenant public cloud is proposed. This chapter explain all the components of the proposed architecture and the rationale of all the components. It also explains how the proposed VM sizing algorithm was evaluated.

Finally, in Chapter 7, the thesis is concluded with a summary of the main findings and further research arising from the main findings.

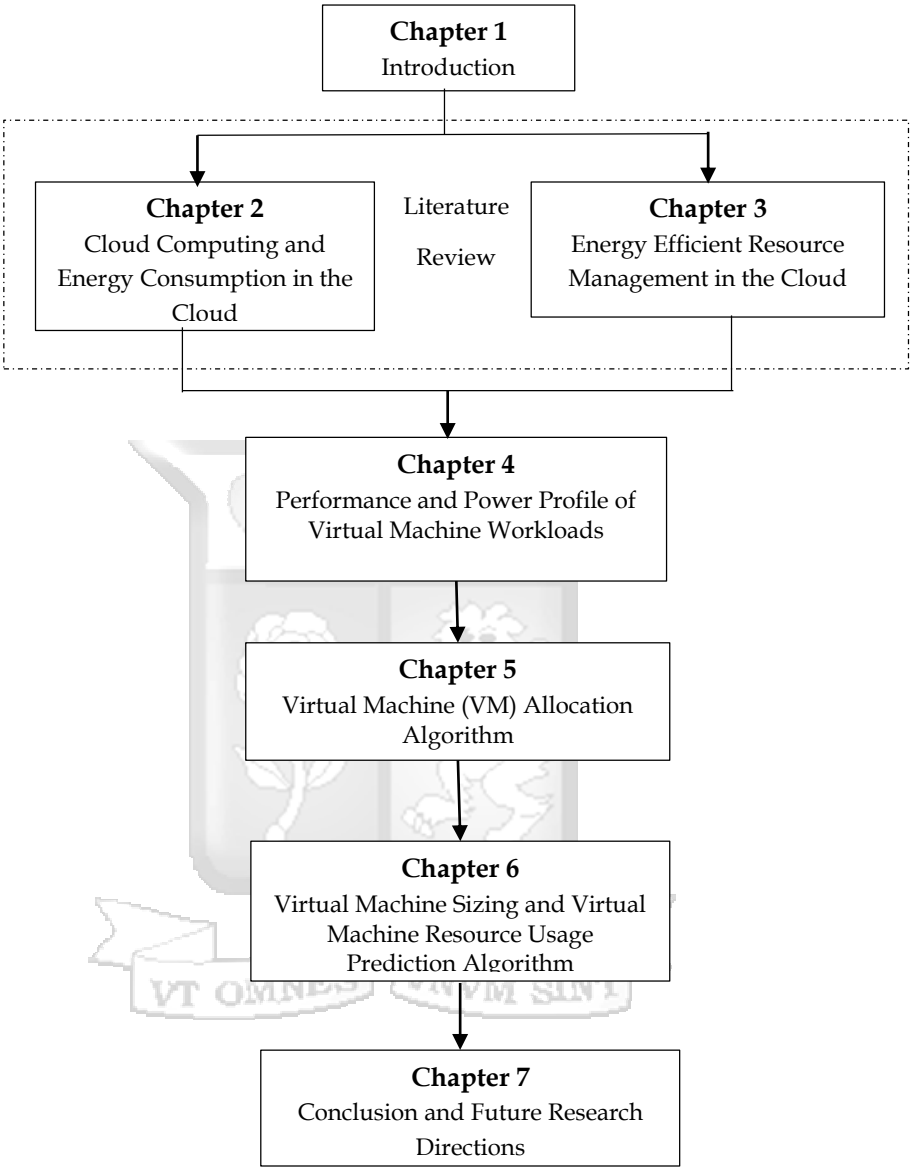


Figure 1.6: Thesis organization

Chapter 2

Cloud Computing and Energy Consumption in Cloud Data Centers

2.1 Introduction

In this chapter, the meaning of cloud computing and its main concepts are explained. Virtualization, which is the main supporting technology in the cloud is also elaborated. Further, energy consumption in cloud data center and data center Servers is explained by identifying the main consumer components of energy. This chapter also explains how power consumption can be measured at host or server-level, the server power consumption model and the factors influencing energy consumption in cloud data center Servers.

2.2 Cloud computing

Cloud computing is a model that provides computing resources on demand or on a rental basis and so users can pay only for resources they use (Sareh, 2016). Therefore, customers can purchase a specific set of resources when they need it instead of renting a fixed amount of physical server, which they may not use all at the same time. The National Institute of Standards and Technology (NIST) (2011, p. 2) defines Cloud Computing as ... *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*.

By shared pool, it means that resources are collected together and then dynamically allocated regardless of their physical location. On the other hand, network access allows the collected resources to be accessed via a network. In addition, rapid provisioning capability allows the service offering to scale so that the changing demands by cloud users are met. Cloud computing allows

applications to be accessed via the internet using a browser, as well as hardware systems and systems software in the data centers that manage user applications.

The cloud model proposed by NIST (2011) has five essential characteristics, three service models, four deployments models and five cloud actors. The essential characteristics include;

1. **On-demand self-service** - this characteristic means that a cloud user can provision computing resources such as memory, processor time, and network access when needed automatically and without the need of involving the CSP.
2. **Broad-network access** - this means that the cloud services are capable of being provisioned over a network and use of heterogeneous thin and thick network clients. For example, services can be accessed through desktops machines, laptops, tablets and mobile phones
3. **Resource pooling** – this means that the CSP pools together a lot of computing resources to give service to multiple cloud users through a multi-tenant model, which allows virtual and physical resources to be assigned and reassigned according to cloud user demands or application task demands. With this model, there is a sense of location independence where cloud users may be unaware of the exact locations of resources but may be at a position to specify a location by way of region. The examples of resources, which are pooled include storage, network bandwidth, processor time and Random-Access Memory (RAM).
4. **Rapid elasticity** - this means that the cloud service offered can elastically be provisioned mostly automatically, to scale outward or inward to fit demand. For the user perspective, the cloud capabilities seem to be unlimited and can be provisioned in any quantity at any time.
5. **Measured service** - this means that the cloud system controls the use of resources by using a metering model (pay-per-use basis). This metering

is applied to each of the services offered. For instance, the use of each resource is monitored, controlled and reported, which provides transparency for both the CSP and cloud user.

In the next sub-chapters, cloud computing actors, cloud computing service models and cloud computing deployments models are explained.

2.2.1 Cloud computing actors

There are five cloud computing actors, which are the people or organizations who interact or participates in a cloud service offering (NIST, 2011, Chaima, 2014). Cloud actors make it easy to understand the operations of a cloud service offering. As shown in *Figure 2.1*, the cloud computing reference architecture, has the cloud computing actors have been defined. These actors are;

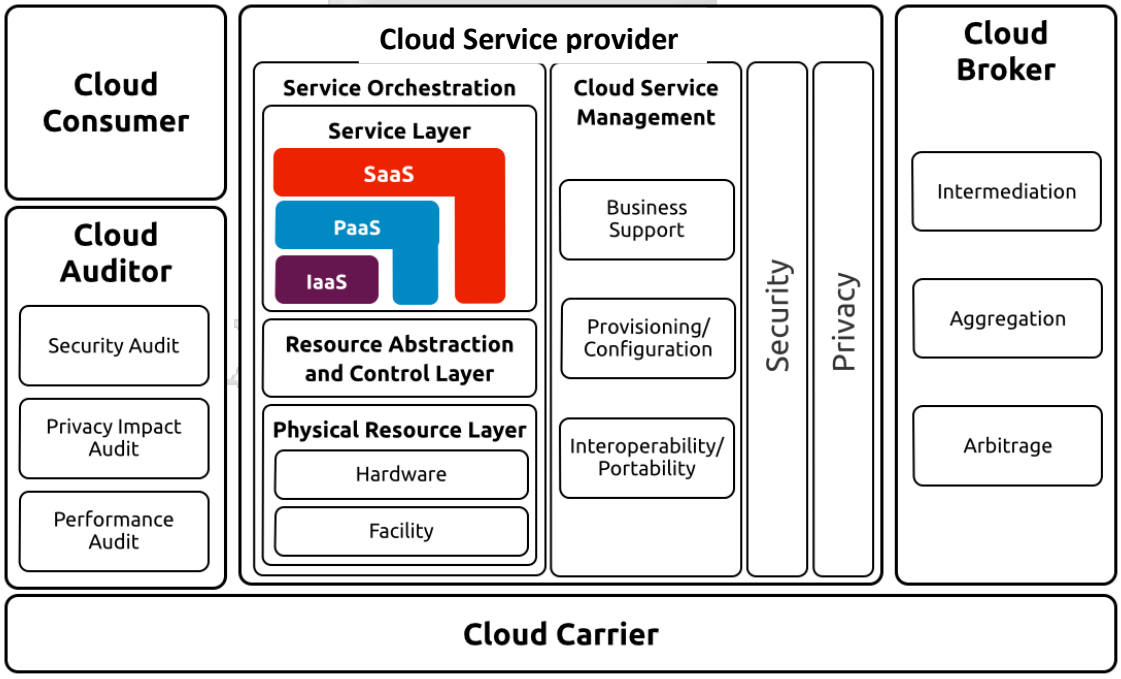


Figure 2.1: cloud computing reference architecture with defined cloud computing actors (Adopted from Vyom, 2013)

Cloud service provider (CSP)

CSPs is the owner of the cloud service. A CSP builds the requested software/platform/ infrastructure services, manages the technical infrastructure

required for providing the services, provisions the services at agreed-upon service levels, and protects the security and privacy of the services. This responsibility may differ depending on the cloud service model – IaaS, PaaS and SaaS. CSP provides service to cloud users to make profits.

Cloud user

Also known as, cloud consumer, this actor uses the services offered by a cloud provider and pays for them. A cloud consumer represents a person or organization that maintains a business relationship with and uses the service from a cloud provider. A cloud consumer browses the service catalog from a cloud provider, requests the appropriate service, sets up service contracts with the cloud provider, and uses the service. The cloud consumer may be billed for the service provisioned and needs to arrange payments accordingly. Like CSP, the responsibility of a cloud user differs depending on the cloud service model – IaaS, PaaS and SaaS

Cloud broker

The cloud broker sits in the middle between the consumer and the CSP. Their role is to help the consumer to overcome the complexity of choosing a CSP. This actor may assist the consumer to combine the features of multiple cloud providers. In most cloud computing setups, the cloud broker submits user computing jobs on behalf of the cloud user and is delivered a software component in a cloud architecture.

Cloud carrier

A cloud carrier is an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers. Cloud carriers provide access to cloud consumers through a network, telecommunication, and other access devices. Cloud consumers can access cloud computing from cloud providers through network access devices, like computers, laptops, mobile phones and tablets. Ordinarily, a CSP will set up Service Level Agreements (SLAs) with a

cloud carrier to provide services consistent with the level of SLAs offered to cloud consumers.

Cloud auditor

A cloud auditor is an actor whose roles is to carry out an independent assessment of the security and quality of cloud services implementations and IT operations. For instance, if a cloud provider hosts card payment operation in e-commerce, the cloud auditor is required to perform Payment Card Industry data Security Standard (PCI DSS) compliance assessment.

2.2.2 Cloud computing service models

A service model explains how the different services offered by the cloud is made available to the user. The services provided by cloud computing can be categorized into three classic layers - Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) (Neha & Rishabh, 2015). Figure 2.2 shows the different services that are accessible in the different layers.

2.2.2.1 Infrastructure as a service (IaaS)

IaaS is the lowest layers and delivers computing resources such as processor, memory, network and storage (Mallavarapu, 2012). In IaaS cloud, users provision VMs and independently run applications with mixed workloads without any control from the cloud provider. As compared to PaaS and SaaS, IaaS users have the most control on the cloud infrastructure and can thus build their applications from scratch. From the CSP point of view, applications are a black box host in a VM. Users get access to the hardware so that instead of investing in on their own infrastructure, which is expensive, organizations are able to rent and pay only for the resources they need (Chaima, 2014).

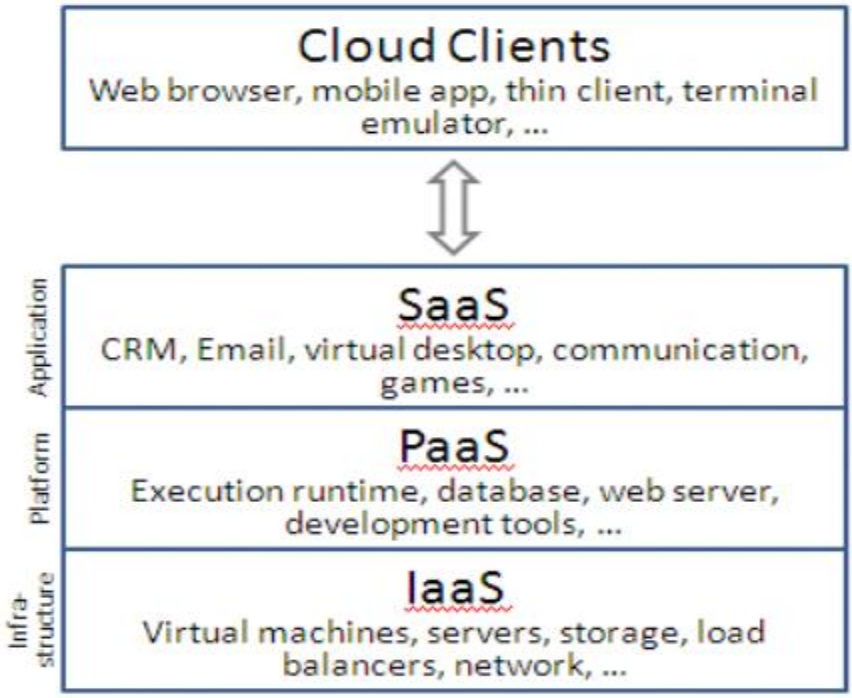


Figure 2.2: Cloud computing service models (Adopted from Gorelik, 2013)

Getting access to IaaS infrastructure is easy and quick as it takes a matter of a few minutes or hours. The reason for IaaS's preference is that it enables an organization to move their physical infrastructure to the cloud while retaining the desired control similar to the one they have in their on-premise private data centers (Mallavarapu, 2012). IaaS model has the closest resemblance to the traditional on-premise data centers and it also has the lowest entry barrier as compared to the other service models (PaaS and SaaS). Because of these reasons, IaaS is by far the most promising model in providing cloud computing services among small organizations and new entrants in the cloud computing space (Esha, Yumnam and Biju, 2016). Thus, many international CSPs such as Google, Amazon, HP, IBM, Citrix, Rackspace, Microsoft, DigitalOcean, Linode and Vultr are already providing IaaS service (Patel, et al., 2015).

Among the IaaS providers aforementioned, Amazon is the leading provider through Amazon Web Services (AWS). AWS provides an Elastic Compute Cloud (EC2), which is a cloud service that provides an elastic computing capacity

(Bankole, 2013). Elasticity in EC2 means that a VM instance's capacity can be increased or decreased in a matter of minutes. EC2 provides capacity on demand and is a good choice due to cloud resource usage dynamicity. EC2 provides a simple web interface that enables users to launch VM instances with an operating system (OS) of their choice. Users can also resize their VM instances with ease and manage network configurations. Some of the benefits of EC2 include elasticity, the ability of customers to select from a wide range of VM instances, a 99.9% service availability, rich web interface and a per second billing model (Bankole, 2013).

The wide range of EC2 instances has been designed to fit a wide range of use cases (Amazon, 2018). The instance types have different combinations of computing resources (memory, CPU, network and storage). The instance types are categorized into general purpose, compute optimized, memory optimized, accelerated computing and storage optimized. General purpose instances can be used to execute workloads with no special needs in terms of the computing resource. On the other hand, compute optimized, memory optimized, accelerated computing and storage optimized instances are used for workload which is intensive in CPU, memory, Graphics Processing Unit (GPU) and storage respectively. For each category, there are different sizes of VM instances, which is based on the capacity of the instance in terms of the number of VCPUs, memory, storage and network speed. This is illustrated in Table 2.1.

2.2.2.2 Platform as a service (PaaS)

PaaS provides an application development environment, which abstracts the underlying hardware, OS, virtualization technology, databases, and web servers (Chaima, 2014). Essentially, PaaS provides tools and development environment for applications, which developers need to build, test, deploy and manage the applications without worrying about the underlying technologies. In this service model, the CSP has more control over the cloud environment as compared to IaaS.

Table 2.1: A1 general-purpose amazon's EC2 instance sizes (Amazon, 2018)

Model	vCPU	Memory (GB)	Storage	Network performance (Gbps)
Medium	1	2	EBD-Only	Up to 10
Large	2	4	EBD-Only	Up to 10
xlarge	4	8	EBD-Only	Up to 10
2xlarge	8	16	EBD-Only	Up to 10
4xlarge	16	32	EBD-Only	Up to 10

The most common and important PaaS providers in the market are Microsoft Azure (Microsoft, 2018) and Google App Engine (Google, 2018).

2.2.2.3 Software as a service (SaaS)

SaaS is the highest level of the cloud service model. In SaaS, complete and functioning applications are provided to the user over the internet. SaaS CSP manages all the hardware and software components for the user and as such, cloud users do not need to know any information, such as the location of physical servers, virtualization technology and OS, about the management the cloud infrastructure. Examples of SaaS offerings are social media applications such as Facebook and Twitter, all Google apps including Google documents and customer relationship management (CRM) applications such as Salesforce (Mallavarapu, 2012).

2.2.3 IaaS cloud deployment models

IaaS service model provides the closest resemblance to the traditional on-premise data centers and it also has the lowest entry barrier as compared to the other service models and as such, it is the most promising service model. It is

categorized into 3 classic deployment models, which are; public, private, hybrid and community as shown in *Figure 2.3* (Sareh, 2016).

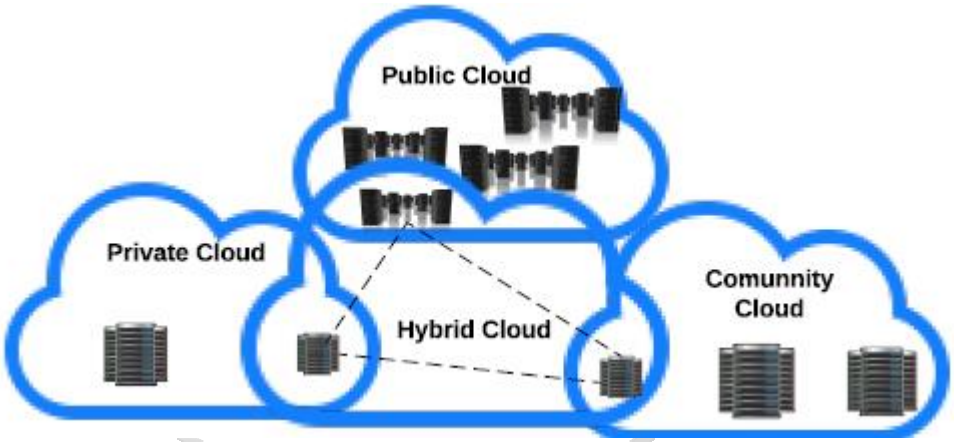


Figure 2.3: IaaS Cloud deployment models: private, public, community and hybrid (Adopted from Sareh, 2016)

2.2.3.1 Public cloud

In a public cloud model, a CSP owns a cloud infrastructure, which is available to the general public. Cloud users subscribe to the cloud service on a pay-per-use basis. Most CSPs, which offer public cloud services also provide a REST-based API, which gives the subscribers the ability to customize and build their own interface for automating certain tasks and to receive communication from the cloud infrastructure when an event happens (Mallavarapu, 2012). For instance, cloud users can automate the process of provisioning and de-provisioning resources to avoid resource wastage and application starving. In a public cloud, many cloud users share physical resources. User application isolation is managed by hypervisors. This concept is known as multi-tenancy – multiple users running their applications in the same physical server. Multi-tenancy is becoming common and can be said to be supported by Bezos' law which states that "The cost of computing will be cut in half every 3 years" (O'Connor, 2014). Because of the reduced cost of executing workloads in the cloud, it is expected that businesses will abandon their private data centers and move to the public cloud as a way of

saving money. Thus, the future of the public cloud is bright. For this reason and as aforementioned, many CSPs such Google, Amazon, HP, IBM, Citrix, Rackspace, Microsoft, DigitalOcean, Linode, Vultr, Safaricom and Angani are already providing IaaS public cloud (Patel, et al., 2015). Angani and Safaricom are Kenyan based public CSPs, which has deployed Cloud data centers in Kenya (Angani, 2019; Safaricom, 2019). Angani has also deployed a data center in Tanzania and Rwanda (AllAfrica, 2018). The disadvantage of public cloud is that cloud users do not have any control over the underlying hardware and the security and performance issues as a result of multi-tenancy.

2.2.3.2 Private cloud

Essentially, private cloud is similar to a public cloud only that in a private cloud, the infrastructure is used internally by the organization owning them. The infrastructure in a private cloud is not available to the general public. The main advantage of a private cloud in that the organizations have full control over their hardware and software deployments. Although private cloud may not offer much value to the organization in terms of cost savings, it makes it easy for an organization, when it is ready to move to the public cloud.

2.2.3.3 Hybrid cloud

A hybrid cloud is a combination of two or more deployment types such as private and public cloud. Essentially, a hybrid cloud is used in design to address the downside of private cloud while taking care of the risk associated with public cloud (Mallavarapu, 2012). A use case for deploying hybrid would be where the private cloud is used to handle the normal workload while the public cloud handles excess workload demand.

2.2.3.4 Community cloud

In a community cloud, the service is provided to a limited number of organizations that share some concern such as security and compliance

requirements. Such cloud deployment models are managed by a third party CSP, which offer the service to all participating organizations.

2.2.4 Multi-tenancy in IaaS public cloud

In a public IaaS cloud, users are allowed to pick VM sizes from CSPs' list of available VM types or sizes (Neha & Rishabh, 2015; Carmody, 2018). A VM size is the measure of resources – CPU, Memory and I/O - assigned to a VM (Sareh, 2016). More often, the users overestimate the amount of resources required by their applications leading to resource wastage. Many cloud users share the same physical servers and run applications side by side. This concept is known as multi-tenancy. In IaaS, where the customer is able to provision CPU, memory, storage and network without the ability to control the underlying virtualization and hardware layers, multi-tenancy occurs when many different VMs belonging to different customers claim tenancy in the same physical machine (AlJahdali, et al., 2014). Although multi-tenancy has its own downside such as security and resource contention due to sharing, it has its own main advantage, which is the financial gains from resource sharing and VM mobility. It is difficult to do away with multi-tenancy, thus the problem that needs to be addressed is resource over-estimation by cloud users in public clouds.

In order to determine the actual amount of resources used by a VM, data about a particular VM has to be collected from the virtualization layer and analyzed and this has to be done by the CSP. From a CSPs' point of view, applications running in a particular customer's VM are a black box host in a VM (Jiaqing, Nipun, & Willy, 2010). Fortunately, the CSP has access to the virtualization layer, where they can monitor resource usage for each hosted VM. From this viewpoint, there are many attempts that have been made by a large organization to address the problem of VM sizing. For instance, ParkMyCloud is an online service tailored for Windows Azure Cloud service customers to get VM size recommendations based on their historical VM resource usage

(ParkMyCloud, 2018). Once the customer gets the recommendations, the customer has to manually resize the VM if a need arises. For customers using Google cloud, there is a VM sizing recommendation service, which suggests new VM types in case of underutilization or overutilization of a given resource (Google, 2018). With this implementation, Google cloud collects VM resource usage at intervals of over 60 seconds to avoid capturing of short resource usage. AWS' VM sizing tools, Amazon CloudWatch is another VM monitoring tool, which is poised to be the best but is not free (Amazon Web Services, 2018). With CloudWatch, one can collect and access all performances and operational data in form of logs and metrics from a single platform for purposes optimizing performance, managing resource utilization and understanding system-wide health status. Amazon also provides other ways of enabling its customers to resize their VMs such as AWS Trusted Advisor. All the methods provided by Azure, Google and AWS cloud services have to be manually completed by customers and seems to fit customers who already have knowledge in cloud computing. Moreover, the customer has to choose from the VM type preset by the CSP for resizing their VMs. Nevertheless, if the recommendations are applied correctly and consistently, cloud customers can save up to 70% on the monthly bill.

2.2.5 Virtualization in cloud computing

Virtualization is the main technology backing up cloud computing and it is based on physical resources abstraction in a way that several virtual resources are multiplexed on a physical one (Sharma, 2014). Virtualization provides high resource utilization, as compared to traditional computing, flexible, elasticity. This makes it possible to run multiple services or applications in the same Physical Machine (PM) including operating systems. A server is divided into number small servers known as Virtual Machines (VMs), which can run different applications independently and a VM can be moved from one PM to another as shown in *Figure 2.5*

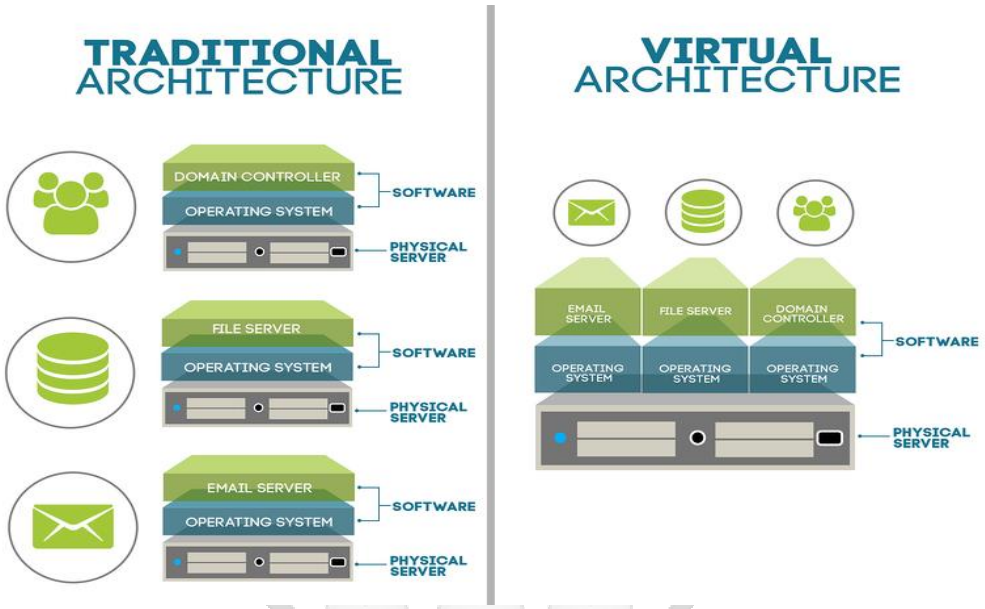


Figure 2.4: Traditional server provisioning and Virtual Server provisioning (Adopted from Live Consulting, 2016)

The hypervisor or Virtual Machine Manager (VMM) is a software layer, which induces the partitioning capability and may run directly on the hardware or on a host operating system (Sharma, 2014). The VMM is responsible for managing the physical resource. A host machine is a PM in which a VMM runs. Examples of VMMs are Xen, VMWare, Microsoft's Hyper-V and KVM. A VM is a representation of a real machine using a software, which provides a virtual operating environment in which an operating system runs. A VM is referred to as a guest machine and it runs a guest operating system. As illustrated in Figure 2.4, virtualization, unlike traditional computing, can be used to run different applications hence solving the problem of computing resource underutilization.

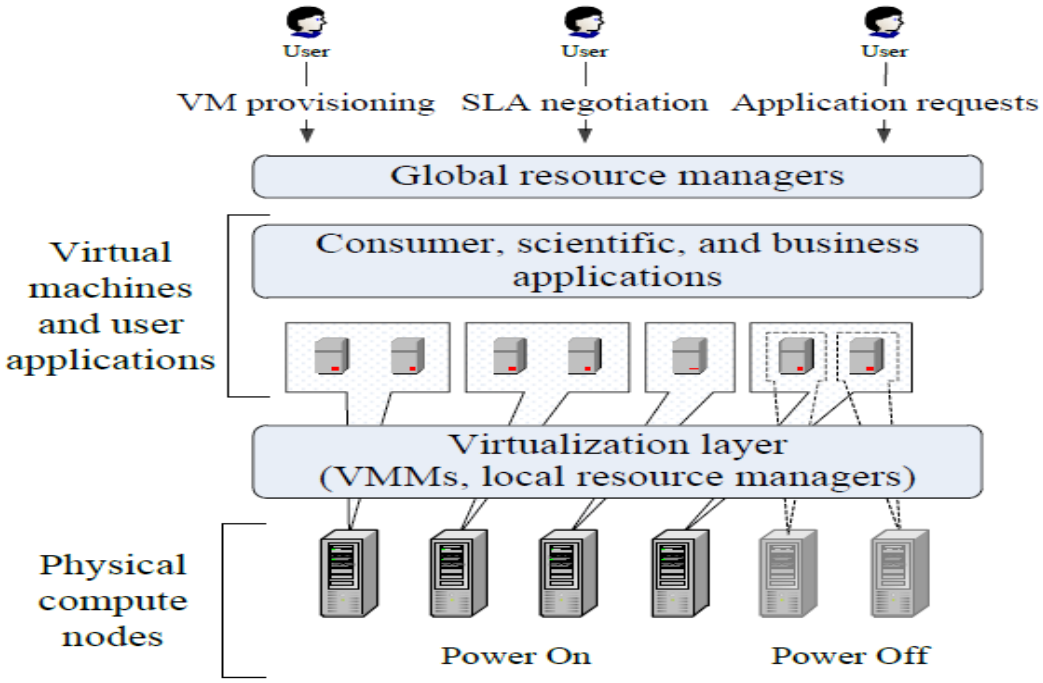


Figure 2.5: Cloud computing high-level system view (Adopted from Anton, 2013)

In computing, virtualization may refer to many things - server virtualization, storage virtualization and network virtualization (Chaima, 2014). This study focus on server virtualization. Another advantage of virtualization is hardware independence. In this regard, VMs are very independent of the underlying hardware. For example, one can configure VMs with CPU and memory, which are different from the PM and sometimes, total resource allocation for VMs may exceed those available in a PM – over-commitment. This also explains why VMs of the same PM can run different operating systems independently.

Apart from improving resource utilization on PMs by consolidating VMs, VM migration is another capability of virtualization. In this regard, a VM can be moved from one PM to another. The widely used type of VM migration is live migration. It enables moving running VM and applications to another PM without

disconnecting the VM from CPU, memory, storage and network connectivity hence zero downtime. With proper live migration, the process may go unnoticed from a cloud user perspective. Although the energy consumed during live migration is not negligible, this technique has been used in VM consolidation for overall data center energy savings (Lidin & Mohamed, 2014). Additionally, virtualization reduces overall data center energy consumption by reducing the amount of hardware in use such as servers (Anton, 2010). This is possible because many servers can be consolidated in one physical server and still provide the required QoS.

There are two types of hypervisors – *bare metal hypervisor* and *hosted hypervisor* (Chaima, 2014). *The bare metal hypervisor* is the one, which can be installed directly on the hardware and controls all the physical computing resources. This hypervisor then shares the computing resources among all the VMs running in that particular hardware. With a *hosted hypervisor,* an operating system is first installed in the hardware, which then hosts the hypervisor that is used to install multiple VMs or operating systems. In both cases, the running Operating System (OS) communicates with the hypervisor and not directly with the hardware. The commonly used hypervisor type in data center is bare metal.

The two main techniques used to implement virtualization is *full virtualization* and *para-virtualization*. *Full virtualization* provides a total abstraction of the underlying hardware so that a guest operating system is unaware of the virtual environment. Thus, the guest operating system executes in a similar way it would execute in real hardware. Microsoft virtual server and VMWare hypervisors provide full virtualization (Sharma, 2014). *Para-virtualization* provides the guest operating system with a similar but not identical abstraction as full virtualization. The guest operating system is made aware of the virtual environment and it provides a performance that is close to executing in physical hardware.

From the foregoing, it can be noted that virtualization is closely related to cloud computing, but they are not the same. Virtualization is just a layer in cloud computing stack but with a major role in the capabilities of cloud computing infrastructure. Apart from the hypervisor, deployment of cloud services, especially IaaS, requires additional software known as Cloud Management Software (CMS) for its orchestration. CMSs available today are OpenStack, Eucalyptus, Opennebula and CloudStack - all are open source and have been used widely to deploy IaaS cloud (Jyoti, Shobha, & Anala, 2015). However, OpenStack has stronger user support because it is feature rich and thus preferred.

In IaaS, virtualization allows many VMs to be host in the same physical server. These VMs belong to different customers and it is the intention of the CSP to co-locate many VMs to better utilize computing resources. As the number of VMs sharing the physical resource increase, they compete for hypervisor capacity and the shared physical resources. As a result, interference is caused, which leads to violation of Service Level Agreements (SLAs) (Xavier et al., 2016). A number of studies reviewed by Chen, et al. (2015) have also shown that virtualization and multi-tenancy are the main cause of resource contention. This is because, virtualization creates a layer between VMs and the hardware, which is an overhead.

When resource contention occurs, performance is degraded because applications are forced to run longer than usual. For instance, CPU is affected by the time slices that are allocated to each of the host VM. On the other hand, disk and I/O suffer because of the time spent to move data between the VM, shared memory and the physical resources. According to some authors, performance degradation leads to computing resources running longer and as a result, more energy is consumed (Mar, Lavinia, Orgerie, & Guillaume, 2016; Kurpicz, Sobe, & Felber, 2014). Generally, as the number of VMs in PM increases, the interference is bound to increase. But this depends on the resource type, which is dominantly

used in such VMs. Some researchers have concluded that performance degradation caused by CPU is ignorable (Mohsen, Hadi, & Mahsa, 2011). On the other hand, disk intensive workloads are not consolidation friendly and its performance degradation caused by contention is not ignorable. Besides, computing trends depict a growing gap between CPU and I/O performance and bandwidth. For this reason, it is observed that during data intensive operations, bottleneck caused by storage and memory leads to low CPU utilization (Vasudevan, et al., 2010).

To address the problem of resource contention, various researchers have proposed various solutions. The starting point to address this problem is to understand how cloud workloads behave so that effective consolidation is achieved. Chen, et al. (2015) have proposed a solution called CloudScope, which diagnoses performance interference before scheduling cloud workloads. CloudScope predicts interference likely to be caused by an incoming VM before it is scheduled. CloudScope has analyzed and benchmarked CPU, memory and disk intensive workloads and results show that the prediction achieves a 9% error and improves VMS performance by 10%. Mar, Lavinia, Orgerie, and Guillaume (2016) reports that to address the problem of performance degradation, one has to determine consolidation friendly workload profiles. The authors have carried out a series of experiments and conclude that performance degradation caused by CPU intensive workload is ignorable. On the other hand, disk-intensive workloads are not consolidation friendly. This is attributed to performance overheads caused by a layer of virtualization software. Besides, according to Vasudevan, et al., (2010), computing trends depict a growing gap between CPU and I/O performance and bandwidth. For this reason, it observed that during data-intensive operations, bottleneck caused by storage and memory leads to low CPU utilization. Other authors have concluded homogeneous workloads (cloud workloads that utilize a single computing resource) are detrimental in terms of

interference as compared to heterogeneous workloads (cloud workloads that utilize different computing resources) (Sareh, 2016).

2.3 Energy consumption in cloud data centers and data center servers

The first step towards ensuring energy efficiency in the cloud is to trace and understand how energy is used in the data center. Particularly, one has to understand the components that consume energy in a data center server. The main components in a server, which consume energy is CPU, disk storage, memory and network (Anton, Jemal & Rajkumar, 2011).

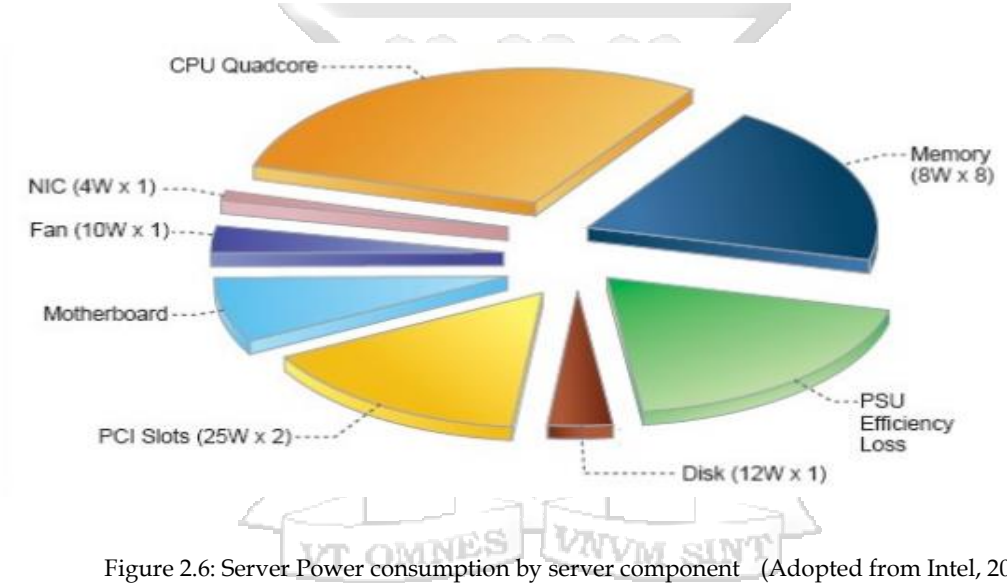


Figure 2.6: Server Power consumption by server component (Adopted from Intel, 2009)

According to Figure 2.6, the CPU consumes the largest portion of energy supplied to a data center server followed by the memory. However, due to improvements in the CPU efficiency and use of DVFS, which enables active low power modes, it no longer dominates power consumption (Anton, 2013). On the other hand, energy consumed by processor greatly depends on processor types. For example, new Intel processor has power saving mechanisms (Intel, 2009). Energy consumed by a data center can be saved up to 50% if VM consolidation is efficiently executed. For example, efficient VM consolidation can ensure VMs are

packaged in the least number of servers so that other servers are shut down thus saving more energy. This is because an idle server consumes 70% of the power when it is fully utilized

Apart from IT load (CPU, disk storage, memory and network), electrical energy is consumed during cooling and distribution. As illustrated in *Figure 2.7*, 33% of data center energy is used for cooling, which is more than 60% of the energy used for real IT load. As the data center servers are used, they emit heat, which needs to be eliminated to avoid additional energy wastage and hardware failure (Jiaqi, et al., 2014). The amount of heat generated is a function of three factors; - frequency and voltage of the integrated circuit, the technology used in manufacturing the components, the efficiency of component design and most importantly, the amount of work done (Intel, 2009). Removing the heat generated allows the component to operate on their safe operating temperature to avoid service degradation or complete damage of the component.

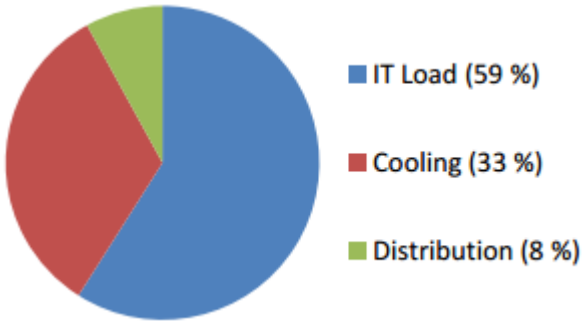


Figure 2.7: Energy consumption by data center components (Adopted from Akhil & Navdeep, 2015)

2.3.1 Measuring Energy Consumption in data center Servers

Data center components which include servers are powered by electricity. The electrical power flows from the external power grid into the internal infrastructure facilities, (IT equipment) through Uninterrupted Power Supplies (UPS) to maintain a consistent power distribution even during possible power failures. A data center environment is made up of two main components, software

and hardware, both of which have been used to measure electrical energy consumed by data center servers.

The hardware-based method of measuring physical server's energy consumption is by using tools such as a smart Power Distributing Units (PDUs), which must be connected between the server and UPS (Chaima, 2014). The PDU is just a smart meter, which measures current at a time interval and multiplies it with voltage to get the watt value. Finally, energy can be obtained according to equation 2.2. Green Open Cloud (GOC) is another example of hardware-based energy measurement framework, which uses sensors to report the electrical energy consumed by physical cloud resources in real-time (Green Grid, 2010). It collects statistics of the power usage in real-time and embeds electrical sensors that provide dynamic measurements of energy consumption of the various computing resources.

Apart from using physical electrical energy meters, attempts have been made to use software-based techniques to measure electrical energy consumed by computers (Smith & Sommerville, 2011). Software-based techniques involve collecting data on server power consumption pattern and analysis, which leads to the creation of a power consumption model that estimates power from server data. Example of data that can be used to generate a power consumption model includes power consumption of each computing resource for idle server, power consumption of each computing resource at different server utilization levels, power consumption of each computing resource at peak utilization, power consumption of each computing resource at different levels of utilization of the different computing resources and power consumption of each computing resource when processing different types of resources.

Measuring energy can be achieved by combining both hardware and software-based method. The software technique is a model embedded in a hardware component. For instance, when carrying out experiments, power usage

can be monitored by executing commands on operating systems such as powerstat command on a Unix based OS. This command is executed on the OS's terminal to collect power reading over time, which are easily converted into energy consumption (Ubuntu, 2018). This command uses a software power meter (on-chip embedded power model), which uses Intel Running Average Power Limit (RAPL) driver. This command works on a mobile personal computer (PC) with a battery as a power source or one that supports the RAPL interface. This means this command can work on a server as long as it supports RAPL. At the end of the run, powerstat returns a number of statistics such average, maximum (max) value, minimum (min) value and standard deviation (SDev).

2.3.2 Modeling server power and energy

Energy and power can be defined in terms of the amount of work a system performs. Energy is the amount of work performed and power is the rate at which a system performs work (Anton, 2013). The units for measuring power and energy are Watts (W) and watt-hour (Wh) respectively. Work is done at the rate of 1 W when 1 Ampere (A) is transferred through a potential difference of 1 Volt (V). A kilowatt-hour (kWh) is the amount of energy equivalent to a power of 1 kW (1000 W) being applied for one hour. Power and energy can be computed as shown in equation 2.1 and 2.2 (Anton, 2013)

$$P = VI, \qquad 2.1$$

$$E = PT, \qquad 2.2$$

Where P is average power consumption, I is current (in Ampere), V is voltage (in Volts), E is energy (in watt-hour) and T is a time (in seconds) interval.

The main power consumption or dissipation parts in metal-oxide-semiconductor (CMOS) circuits comprises static and dynamic power (Mittal, Bajaj, & Neha, 2013). Static power dissipation is associated with inactive logic gates (i.e., not currently switching from one state to another). It is caused by leakage currents

while the gates are idle; that is, no output transitions. Dynamic power dissipation occurs in the logic gates that are in the process of switching from one state to another. It is caused by the current flow from the charging and discharging of parasitic capacitances. From a data center server perspective, static power is consumed when a server is idle and not processing any application workload and dynamic power is consumed when a server is processing application workload.

Estimating energy consumed by a data center needs to be modeled because sensors cannot be used to measure VM energy consumption and that, energy-metering devices cannot be connected to data center servers. Creating an energy and power model for any system such as a server requires that the system is studied well such as by monitoring power consumption under a different scenario. To achieve this, one can make use of the power monitoring techniques inbuilt in modern servers. These techniques collect accurate power usage statistics in a real-time manner.

Fan, Weber and Barrosso (2007) found a strong relationship between CPU utilization and the total power consumed by a server – power consumed by a server grows linearly with the growth of CPU utilization from the value of power consumption in the idle state up to the power consumed when the server is fully utilized. This relation is shown in equation 2.3. Extensive experiments on several thousand nodes using different types of experiments showed that the derived model accurately predicted the power consumed by a server with an error of about 5%. The power consumption behaviors in their experiments were found to exhibit the characteristics shown in *Figure 2.8*.

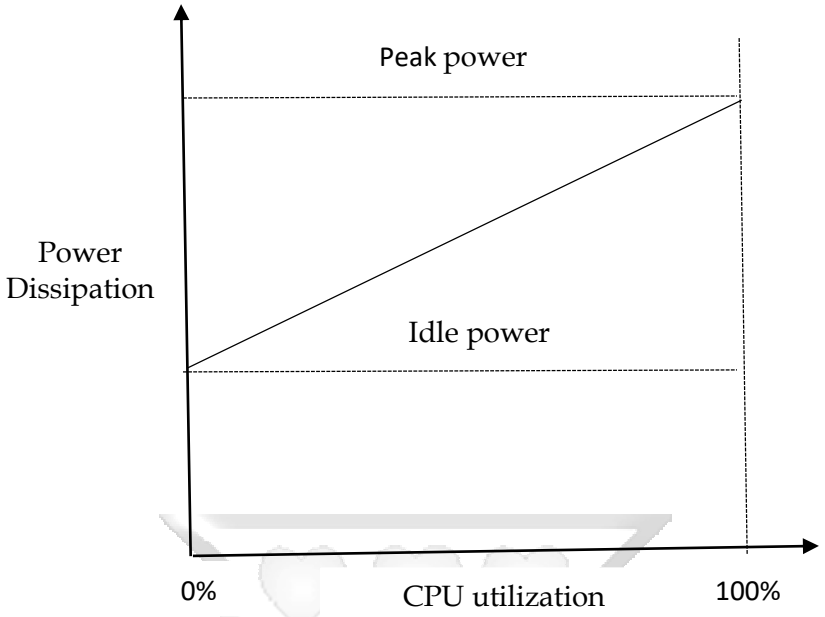


Figure 2.8: The relation between power consumption and CPU utilization of a server (Adopted from Chaima, 2014)

$$P = P_{idle} + (P_{Peak} - P_{idle})*U \qquad 2.3$$

where P is the estimated power consumption, P_{idle} is power consumption by an idle server (related to Static power dissipation), P_{Peak} is the power consumed by the server when it is fully utilized, and U is the current percentage CPU utilization. $P_{Peak} - P_{idle}$ is the dynamic power (related to Dynamic power dissipation). This model considers only the CPU because it is the main component which consumes energy in the server and has a wider dynamic power range. Form their experiments, it was found that an active server with very low CPU utilization still consumes 50–70% of the power it would consume when fully utilized due to idle power and this is contributed by all components' idle power (Sampaio, 2015). The dynamic power ranges for the other components are narrower: < 50% for DRAM, 25% for disk drives, and 15% for network switches thus ignoring them in the model does not create a significant difference. The presented simple power model enables easy determination of server power usage given CPU utilization and

power consumption value of idle server and fully utilized server. More complex power models have been proposed, which includes other components – memory, network access and hard disk access (Dhiman, Mihic, & Rosing, 2010; Basmadjian et al., 2011).

2.3.3 Data center energy efficiency metrics

The Green Grid (2007) has offered two data center power efficiency metrics: Power Usage Effectiveness (PUE) and data center Efficiency (DCE). PUE is defined as the total power consumed by the data center divided by the power used by the IT equipment, as shown in equation 2.4.

$$\text{PUE}=\frac{\text{Total Facility Power}}{\text{IT Equipment Power}} \qquad 2.4$$

DCE is the ratio of IT data center energy efficiency and is defined as the reciprocal of PUE as shown in equation 2.5

$$\text{DCE}=\frac{\text{IT Equipment Power}}{\text{Total Facility Power}} \qquad 2.5$$

$$\text{DCE}=\frac{1}{\text{PUE}} \qquad 2.6$$

However, these two metrics do not communicate a lot of information because they simply measure how much of the energy entering a data center facility is used to power the computing devices within, versus the amount used for cooling and overhead of the facility. Another performance metric that can be used to measure and rank the energy efficiency of servers is Performance per Watt (PPW) (Chaima, 2014, p.21). PPW can be defined as "*…a measure of the energy efficiency of a computer architecture or computer hardware. It can be represented as the rate of transactions or computations or a certain performance score that can be delivered by a computer for every*

watt of power consumed". This metric is important because it ranks servers regardless of their architecture, manufacturer or size. For instance, PPW of a CPU can be computed according to equation 2.7. The higher the PPW value, the higher the energy efficiency of the server.

$$PPW_{cpu} = \frac{I}{P} \qquad 2.7$$

where PPW_{cpu} is the PPW of a CPU, I is the number of instructions processed by the processor per second in Million Instruction per Second (MIPS) and P is the CPU power.

2.3.4 Factors influencing energy consumption in data centers servers

To improve energy efficiency, it is imperative to find out the factors, which influences the amount of energy consumed by a data center and data center servers and hence the causes of energy wastage in cloud data centers and data center servers.

2.3.4.1 Level of server utilization

Server utilization is the percentage of time during which a server is busy processing application workload and it depends on how workload patterns vary from time to time (Anton, 2013). Low server utilization is a major cause of energy wastage and is caused by the inefficient utilization of computing resources (Chaima, 2014). At high server utilization, computing resources are efficiently used and as a result, less physical servers are used hence saving energy that would have been used by powering more physical servers. Generally, the level of server utilization determines how well energy is utilized in a server. Natural Resources Defense Council (NRDC) (2014) reports that average server utilization for small-to-medium data centers , with market segmentation by electricity consumption of 49%, is 10%, and 50% for High-Performance Computing (HPC) data centers , whose market segmentation by electricity consumption is 1%, and that the physical machines drew up to 90% of their peak power. Clearly, this is resource

over-provisioning, which leads to increased energy consumption because many servers have to be used.

A six-month data analyzed from about 5000 servers revealed that, although servers are generally not idle, their utilization never reaches 100% (Anton, 2013). According to an analysis conducted by Dabbagh et al (2015) on Google data center resource usage, 65 % of CPU and 45 % of memory goes to waste. This shows that application workloads utilize fewer resources than what is provisioned. With high resource utilization, the number of physical servers required will be greatly reduced thus reducing the amount of energy used in data centers.

Moreover, slim dynamic power ranges of computing resources cause low server utilization because an idle server consumed up to 70% of its peak power (Sareh, 2016). In this regard, it makes sense to operate at high server utilization levels. However, according to Delimitrou (2015), there are three main challenges towards ensuring that servers are fully utilized at 100% all the time. These challenges are; diurnal patterns experienced on server workloads and load spikes, which calls for resource over-provisioning leaving servers underutilized, servers are heterogeneous and have changing configurations, thus matching diverse workloads to the servers is not trivial and at high server utilization, there is interference due to resource contention leading to performance degradation. Particularly, interference has diverse effects on QoS, especially on compute-intensive workloads.

2.3.4.2 Idle energy wastage by servers

An idle server can consume over 70% of their peak energy hence it could potentially be turned off to save energy (Chaima, 2015; Anton, 2013). This behavior of servers does not represent any proportionality in an increase in energy consumption with respect to system throughput. As a result, a server running at 20% can consume 80% of the energy consumed by a server operating at 100%

(Mastelic, et al., 2015). This represents a huge energy loss when servers run idle without any throughput and is usually the case for many typical servers. In this regard, one can see that this is a cause of idle energy wastage. Moreover, if an application workload does not utilize computing resources in a balanced manner, the idle components will also waste idle energy (Mirabel & Siddiqui, 2015). For example, if an application workload is CPU-intensive, then memory idle energy goes to waste. Therefore, it is essential that co-located VMs utilize all computing resources in a balanced without leaving some to idle. This is where heterogeneous workloads come in handy.

NRDC (2014) reported that as the number of servers in a data center continues to grow, so is the number of comatose servers. A comatose server is a server that is powered and uses electrical energy without delivering any useful IT service. Such servers may have been left when a certain project ended or a business process changed and since then, the servers were not removed or no one is tracking them. According to Uptime Institute (2013) cited in NRDC (2014), an estimated 20 to 30 % of all servers in large data centers are idle, unused or obsolete but still consume energy. The main causes of the rise of comatose servers in data centers are; - 1) Lack of focus such as not budgeting time for staff to identify and remove comatose servers and 2) Aversion to risk such as Information Technology (IT) managers fear that, by removing any previously installed servers, they may interfere with application functions that occasionally run on the servers.

2.3.4.3 Adoption of energy efficient solutions in data center servers

Natural Resources Defense Council (NRDC) (2014) reports that it is only large Cloud data center that have adopted energy efficient data center practices. Large cloud data centers such as those owned by Google can have to 80, 000 – 100, 000 servers each with 8-core dual-threaded process, 16 GB and 1 Terabyte (TB) (Barroso & Hölzle, 2009). Such data centers cover around 100,000 sq ft. of space and can consume as much energy as 25, 000 households in the United States

(Dayarathna & Wen, 2016). Alas, the large Cloud data center accounts for only 5 % of global energy consumption. The rest, 95 %, is left to small and medium firms, which are terribly energy inefficient because of the lack of adoption of energy efficient solutions and practices. Such solutions and practices include server and network consolidation, data center wide thermal management, purchasing and installing energy efficient hardware to replace old hardware, power planning and management (such as checking from time to time to identify and remove comatose servers) and installation of energy management software (NRDC, 2014).

Although rising energy costs is an incentive to adoption of energy efficient practices, the pressure to keep up technological advancements have made many organizations to treat energy efficiency with low priority (NRDC, 2014). This has led to organizations not adopting even simple and cost-effective power management software, which can monitor measure and manage both hardware level and software level energy usage. For example, energy management software offered by TSO Logic is relatively affordable and can measure data center power demands, active and comatose servers and energy cost, as well as show how these change over time and assist in relocating application workloads and shutting down servers (TSO Logic, 2017). Nevertheless, some data center operators feel that by adopting automated energy usage monitoring, their employment is threatened and thus they discourage its adoption (NRDC, 2014). Moreover, power-saving features embedded in hardware, which can monitor hardware utilization and report to data center dashboards, are often disabled because of the perceived management complexity and risk associated with switching off servers. In this regard, even organizations running full-scale cloud clusters do not deploy energy management solutions.

In addition, cloud providers have poor habits of procurement, which includes focusing on initial cost rather than Total Cost of Ownership (TCO) (NRDC, 2014). When a procurement procedure focusses only on initial purchase

rather than long-term electricity costs, it may miss on energy efficient equipment in the market. For example, Subramaniam and Feng (2013) report that with the arrival of Intel's Sandy Bridge and Standard Performance Evaluation Corporation Power (SPECPower) benchmark, energy-proportional computing is achievable hence energy consumption by servers at idle state and low utilization can be reduced. Furthermore, NRDC (2014) highlights that 80 % of IT departments in most cloud service providers do not pay their power bills (finance department does) and so they do not see the need to make data center energy efficiency a priority. In addition, the IT department does not see any incentive for implementing energy efficient practices because they are not evaluated based on the amount of energy saved. In fact, IT staff have no access to power bills and most of them are more concerned with software costs. This is in spite of power bills taking the lion's share of operating costs as seen earlier in *Figure 1.1*. This division of accountability and split incentives are a barrier to the adoption of energy efficient solutions.

2.3.4.4 Server utilization metric

Server utilization metric is the unit of measure of the percentage of time during which a server is busy processing workload tasks (Anton, 2013). Lack of a common standardized server utilization metric has been a cause for energy wastage for many decades (NRDC, 2014). Increasing server utilization offers the best option for improving data center IT energy productivity as compared to PUE and Power Supply Efficiency (PSE) (Sanjeev, 2015). In fact, below 50 % server utilization, a continued increase in server utilization offers the highest energy usage productivity because of efficient use of the idle energy (Delimitrou, 2015; Mastelic, et al., 2015; Sareh, 2016; Anton, 2013).

For many years, CPU utilization has been the measure of server utilization but it is not the best since different application workloads have different CPU intensities with some of them being memory, network or I/O intensive than CPU-

intensive. Besides, CPU shows the amount of work with no way of determining if that work is useful or otherwise (NRDC, 2014). As a result, a number of new metrics have sprung up to take care of other data center parameters. For example, the Green Grid (2010) developed a metric based on data center design, executing software, data center hardware, CPU, memory and disk as parameters. The Green Greed (2011) also developed another metric, which attempts to measure server utilization at the application level, for example tracking the number of emails sent by a server.

Other metrics include Power to Performance Effectiveness (PPE), which measures server performance per kilowatt, and SPECpower_ssj2008v1.12, which provides a means to measure power in conjunction with a performance metric. Unfortunately, there is slow adoption of these metrics because they are complicated to implement and cannot deliver complete reports on their own without the need for multiple implementations. Furthermore, different server designs have different levels of energy efficiency hence cannot work across all server designs. Therefore, average CPU utilization and average data center utilization (average server utilization when not in sleep mode over a period), will remain in use until better metric is developed.

2.3.4.5 Data center thermal management

Thermal management is made possible by the use of cooling units and fans, whose actions are controlled by the ambient temperatures in a data center environment (Villebonnet & Georges, 2014). Any increase in temperature would cause an increase in cooling energy. Therefore, the amount of heat produced by a server is an important consideration in managing energy usage in data centers. As illustrated in Figure 2.7, 33 %of data center energy is used up by the cooling unit, which is more than half of that used by IT load (about 59 %). Villebonnet and Georges (2014) have proposed a thermal-aware algorithm whose aim is to maintain uniform server temperature by maintaining a uniform load across

servers of a cluster. According to the algorithm, server temperature should not exceed the server's threshold set temperature and if it does, VM(s) is migrated to another server. In addition, the algorithm detects server under load to ensure maximum server utilization. In summary, the authors' idea is to reduce server hot spots (server with peak temperature) in a data center by sharing heat production, which occasionally forces the cooling units to cool the entire system, including the 'colder' servers.

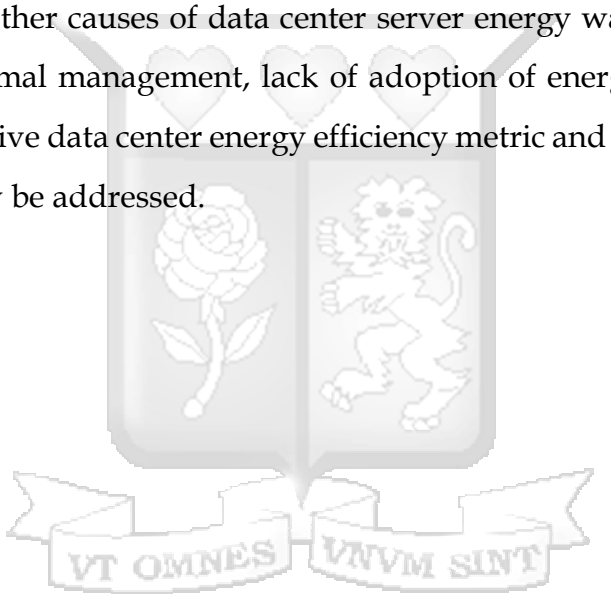
In a review, Neeraj, Manpreet and Sanjeev (2016) describes a thermal-aware resource allocation technique proposed by Al-Qawasmeh et al (2015) in which energy consumed by the cooling unit is reduced by ensuring that individual tasks are completed by their set deadlines. The objective is to determine a performance state within a server, which results in less energy consumption by the cooling while completing a workloads task by its individual deadline. The authors have reported a 9 % reduction in power consumption on simulated experiments.

In addition, to reduce energy consumed by the cooling unit, some cloud computing providers have located clusters in cold geographical locations and undersea to benefit from free cooling such as Microsoft (Microsoft, 2016) and Google (Oxford Research, 2015). According to Microsoft (2016), putting a data center under the sea not only derives a benefit of free cooling but also has a logistics advantage because many people live close to the sea and that clean energy can be generated from sea waves to be used in the data center. Google has also successfully deployed data centers close to sea such as Google Hamina plant and others in Finland to benefit from cool climate (Oxford Research, 2015).

2.4 Summary

This chapter describes the characteristics of cloud computing and its supporting technologies such as virtualization. Because of the reduced cost of executing workloads in the cloud, it is expected that businesses will abandon their private data centers and move to the public cloud as a way of saving money. This

means that more energy will be consumed in the public cloud deployments and will need to be addressed. This chapter also discusses data center energy efficiency metrics. It was discovered that among all the data center energy efficiency metrics covered, PPW is very important because it shows how efficiently the energy that goes into processing workload processing is used. Thus, this should be the recommended data center energy efficiency metric. Further, this chapter discusses the factors, which influence energy consumption in data center servers. It was discovered that the level of server utilization, which is generally low, contributes largely to the wastage of energy in data centers. Thus, if it is successfully addressed, the other causes of data center server energy wastage such as lack of data center thermal management, lack of adoption of energy efficient solutions, use of non-effective data center energy efficiency metric and wastage of server idle power can easily be addressed.



Chapter 3

Energy Efficient Resource Management in the Cloud

3.1 Introduction

The purpose of this chapter is to review the classic policies in place for energy efficient management of computing resources in data center servers. It will also cover different research works that have attempted to address the problem of resource management especially in ensuring high levels of resources utilization in data centers with the aim of achieving energy savings. The chapter attempts to uncover the fact that the main problem of energy wastage is low resource utilization due to resource overprovisioning and that this problem is not trivial because of data center *performance unpredictability*. The chapter focuses on three classic power saving techniques and workload consolidation and its role in data center energy efficiency. Finally, this chapter is concluded with a summary.

3.2 Classic Power Saving Strategies in Cloud data center servers

The classic power saving strategies in cloud data centers servers include proportional computing using DVFS, hardware-based energy saving capabilities, server switching and energy-aware workload consolidation.

3.2.1 Power savings using Dynamic Frequency and Voltage Scaling (DFVS)

DVFS is an energy saving technique that is used to achieve computing proportionality. Barroso and Holzle (2007) define proportional computing as energy efficiency technique where the energy consumption by servers is proportional to the amount of workload. In this regard, idle servers should consume no energy. Unfortunately, energy consumption of computing units is un-proportional - when server load is low, energy consumption is still high. To achieve proportional computing, DVFS is used. DVFS is an energy saving

technique in computer architecture that is used to save energy when the server load is low. In this technique, the frequency and voltage of the CPU are scaled dynamically to match the amount of server load. According to this approach, if the server load is at X % of peak load, then the energy consumption should be at X % of peak energy. Dynamic Voltage and Frequency scaling of CPU is applied for improving the energy consumption of the data center. The frequency of CPU is decided according to the workload by the resource controller, which is installed on each server (Kamga, 2013). The power, *P*, of the CPU is computed as shown in equation 3.1.

$$P = V^2 * F * C \qquad 3.1$$

where *V* is the voltage, *F* is the frequency and *C* is the capacitive load on the system (Patel & Bheda, 2014). It is observed that if voltage and frequency are lowered, there will be a significant reduction in power consumption. Kamga (2013) has also shown that applying DVFS in each core of a CPU is possible. If per-core DVFS is to be used, the author proposes that the core frequency f_{core} is computed according to equation 3.2

$$f_{core} = [(f_{high} * t_{high}) + (f_{low} * t_{low})]/((t_{high} * t_{low}) \qquad 3.2$$

where f_{high} the highest frequency, t_{high} the number of occurrences of the high frequency and similarly for the low frequency.

DVFS has been used to build products available in the market such as AMD Turbo Core, Intel Turbo Boost, and Intel Enhanced Speed Stepping Technology to reduce energy consumption according to workload. Patel and Bheda (2014) have used the concept of DVFS in live VM migration. Their proposal involves monitoring CPU utilization, DVFS adjustment, and real-time migration. They report a reduction of execution time and energy consumption as a result of DVFS application. Unfortunately, they note that this method has a limitation when the number of VMs in a PM increases and that the recent improvements in memory

and processor technologies have saturated clock frequencies and have introduced better sleep modes (Hu et al., 2015). Moreover, DVFS is hardware-based technique and works well only on CPU bound tasks because dynamic power ranges for other computing resources (memory, disk and network), as compared to their static power, are much narrower (< 50% for DRAM, 25% for disk drives, and 15% for network switches)(Anton, 2013).

Tang et al. (2015) present DVFS-enabled Energy-efficient Workflow Task Scheduling algorithm (DEWTS) tool, which uses DVFS. Conducted experiments by authors report a 46.5% energy savings. Conversely, as DVFS is too dependent on the hardware, the resulting energy savings are low compared to other methods. Although DVFS is a good solution, its savings are small because an idle server will still consume over 70% of peak energy.

Apart from DVFS, another technique for managing dynamic resource management is dynamic memory allocation known as *Memory Ballooning* (VMware, 2015; Bronson, P., & Raja, 2018). This technique allows the hypervisor to reclaim unused memory from one VM to share it with another. This technique allows the memory needs for VMs to exceed that available in the host thus achieving a higher hypervisor consolidation ratio. However, for IaaS multi-tenant clouds, this may not be fair because each customer pays for resources they requested and thus such resources must be available at all times. Even if multi-tenancy is eliminated by using stateless servers, data storage would always need multi-tenancy (Perera, 2016). With these techniques, users do not need to overprovision memory, which will otherwise be used for file system caches (VMware, 2015).

Because of the observed failures of DVFS, powering down or switching off servers when they are not in use is a viable option (Anton, 2013; Chaima, 2014; Jiaqi, et al., 2014).

3.2.2 Server power switching

Switching off unused or idle data center servers can result in significant power savings because an idle server consumes up to 70% of its peak energy (Zhang, et al., 2012). The servers would then be turned on when needed. For this reason, server shutdown techniques have been developed to keep the number of powered servers in line with the actual workload. However, deciding which server to switch off, when to do it and for how long is a complex process, which calls for careful planning and thus data center administrators have not embraced this technique (NRDC, 2014). Another reason why this technique has not been embraced is that, until recently, servers were not designed to be switched off and that switching off and switching then on later consumes energy and takes time. Therefore, data center administrators shy off for fear of interrupting with services, potential hardware failure and inability to quantify energy gains versus the loss of service quality due to long booting time.

According to Rais, Orgerie and Quinson (2017), shutdown techniques require that data center hardware have the ability to remotely switch off and on servers and that energy-aware algorithm should utilize this ability in a timely manner. The implementation of Advanced Configuration and Power Interface (ACPI) has five sleep states on the Linux kernel- suspend to idle, standby, suspend-to-RAM, suspend to disk and system shutdown state. However, for data centers, only suspend to idle and system shutdown state are available for use. Suspend to idle allows all user spaces to get frozen and all I/O devices put into low energy states. System shutdown shuts down the system completely such that the system has no memory state and is not performing any task. With this option, the server consumes no energy and requires a total boot to bring it on.

Orgerie, Lefevre and Gelas (2008) cited in Rais, Orgerie and Quinson (2017) proposes a shutdown strategy by considering the time a server is idle and using

this time to decide whether a server should be switched off. The authors define a time T_s, which is a time threshold such that, if a server is idle for less than $T_{s,}$ then it should remain idle to save energy. Additionally, T_s should be greater than the total time the server takes to switch off and on. According to the authors, there are two ideal shutdown policies: *knowing the future* and *aggressive shutdown*. Knowing the future has it that the future dates and lengths of the idle period are known for each server. On the other hand, aggressive shutdown posits that a server is shut down immediately it is idle without any prediction attempt. Although both policies have been found to save energy, the latter consumes more energy because idle server periods may be less than T_s. Although shutdown techniques save energy, they cannot be used in isolation. For instance, if servers run at peak load consistently, energy savings from this technique will not count. Thus, other techniques such as energy-aware workload consolidation, which is discussed in detail later, need to come into the picture. In this regard, Gokul and Priya (2016) have proposed a dynamic consolidation technique whose objective is to reduce idle power wastage and improve performance. The authors have observed that there were an energy cost and performance overhead during server startup. In this regard, however, if a server has no task to process, it is not switched off. Instead, it is put into an idle state for a while before it switched off during which an assessment is done to find out how long it will take for the machine to be useful again. This is to ensure that energy consumed by shutting and booting the machine does not exceed the energy consumed by the idle server.

3.2.3 Energy savings hardware capabilities

A server is made up of many components such as CPU, memory, fans, power supply and disks whose power consumption efficiency can be improved by providing hardware optimization (Anne-C´ecile et al., 2014). For instance, in High-Performance Computing (HPC), servers frequently access storage disks thus consuming more energy. However, by spinning down the disk platters, less

energy is consumed. Generally, the objective is to reduce disk access so that the disks are spun down as long as possible. In this regard, Hard Disk Drives (HDDs) are being replaced by Solid State Drives (SSDs), which is becoming increasingly affordable and consume less energy. There are a number of ways in which SSDs assist in reducing power usage. SDDs utilize flash memory and do not have any moving parts, which would consume energy. Thus, SDDs are ideal for high-density VM environments because they provide high-speed and consistent access, which results in less time spent in storage access operations. In this regard, the use of SSD enables a PM to hold many VMs without loss of performance (Intel, 2016). Furthermore, SSDs require 79% less power for cooling as compared to traditional HDDs. All these benefits result in energy saving in data center servers. In addition, manufacturers can allow individual server components to go to sleep mode independently when they are not in use (Anne-C´ecile et al., 2014). For example during computing phases, Network Interface Card (NIC) can be put to sleep state since they may not be required. Manufacturers can also increase frequencies, speeds and voltages available to a component making it able to adjust to current load in what is termed as proportional computing.

Moreover, Intel announced that the design of their new processors could be in favor of energy efficiency over speed, which technically calls to an end of Moore's Law (Anderson, 2016). Prior to this, Intel had produced the core M series processors, which are 50 % faster in computation speed, 40 % faster graphics performance and 20 % longer battery life (Team, 2014). However, this family of processors has not been used in commercial servers. On the other hand, AMD has developed an Accelerated Processing Unit (APU), which is formally a CPU and Graphics Processing Unit (GPU) on a single chip. The aim of this design is to reduce energy consumption and it has helped reduce energy consumption by between 10 and 20 % (AMD, 2017).

From a hardware perspective, Kurpicz, Sobe and Felber (2014) have carried out experiments to study the differences in energy consumption in various cloud workloads on different hardware and architectures. They have used various parameters such, processor manufacturer, processor mode, processor core and threads, memory size, hard disk characteristics, to vary the hardware and architecture. They have shown that some architectures are energy efficient on CPU-intensive workloads while others are efficient on disk-intensive workloads.

Although the deployment of energy-efficient hardware is a crucial step, getting rid of underutilized servers is a far more effective approach, which is possible through effective consolidation (Elijorde & Lee, 2015). Effective consolidation can be achieved through monitoring resource utilization by application workloads and ensuring that server resource utilization is generally high.

3.3 Workload Consolidation

Workload consolidation, which is also referred to as server consolidation or VM consolidation is a process which can be divided into four areas, namely, host overload detection, host under-load detection, choosing the right VMs to migrate from under-loaded and overloaded hosts also known as VM selection and VM allocation to under-loaded hosts also known as VM placement (Chang, Gu, & Luo, 2017). Workload consolidation has been studied extensively by various researchers (Anton, Jemal, & Rajkumar 2011; Ashwin et al., 2015; Zhang et al., 2012; Mohsen, Hadi, & Mahsa, 2011; Mar et al., 2016; Delimitrou, 2015; Mastelic et al., 2015). Workload consolidation reduces the number of powered servers by consolidating workloads to fewer servers thus effectively improving server utilization and also shutting down idle servers. This section elaborates on VM allocation and migration, VM sizing and VM workload characterization and

mixing as the components of workload consolidation and their role in ensuring energy efficiency in cloud data center servers.

3.3.1 Energy efficient virtual machine migration and placement

VM placement is defined as placing a new VM in a selected PM whereas VM migration is a dynamic consolidation technique, which involves moving a VM from one PM to another (Khan et al., 2018). To ensure energy efficiency, the problem of VM migration and placement can be divided into three sub-problems, namely; - when to migrate, which VM to migrate and where to migrate (Choudhary, Rana and Matahai, 2016; Anton, Jemal, & Rajkumar, 2011). According to these sub-problems, it is important to know the right time to migrate, the right VM to migrate and the right destination PM in order to be energy efficient. One of the triggers of 'when to migrate' sub-problem is the QoS (Anton, Jemal, & Rajkumar, 2011). In this case, overloaded or under-loaded servers are detected for purposes of VM migration. If VM migration is triggered, the next step is to determine which VM to migrate. In the case of an under-loaded host (PM), all VMs are moved to other PMs and such a host is put to idle mode or shutdown. In the case of host overload, one or more VMs need to be migrated another host(s) until the host's load balances.

Anton, Jemal and Rajkumar (2011) have studied three VM selection policies namely Maximum correlation, Minimum Migration Time and Random selection. Maximum correlation is based on the idea that if VMs have a correlation in terms of resources usage, there is a high probability of server overload. Thus, VMs, which have the highest correlation, are selected for migration. Minimum Migration Time selects a VM for migration if it requires the minimum amount of time to complete the migration process as compared to other co-located VMs. The migration time is estimated by dividing the amount of required RAM by the spare bandwidth available in the overloaded server. In random selection, a VM for

migration is selected randomly. Choudhary, Rana and Matahai (2016) have summarized some of the heuristics for VM consolidation using migration as shown in *Table 3.1*

Table 3.1: Heuristics for VM consolidation using migration *(Choudharya, Rana, & Matahai, 2016)*

Goals	**Server consolidation**	**Load Balancing**	**Hotspot mitigation**
When to migrate?	Cold spots on PMs	Load imbalance on PMs	Hotspots on PMs
Which VM to migrate	VMs from lightly loaded PMs	VMs from overloaded PMs	Bunch of VMs from hotspot-PM
Where to migrate	Higher loaded PMS	Lightly loaded PMs	PM which has enough resources to house

After VM selection, determinants for 'where to migrate' sub-problem include co-located VM interference and the correlation between workloads of co-located VMs and statistical multiplexing. Co-located VM interference is as a result of resource contention where the demand to a particular resource by co-located VM exceeds its supply. If VM interference is ignored during VM migration, it results in performance degradation and potential energy wastage (Fei, et al., 2014). Also known as joint VM provisioning, statistical multiplexing enables a VM to borrow resources from co-located VMs while it experiences peak workloads (Sareh, 2016). Correlation between workloads of co-located VMs is used so as to consolidate VMs with least correlation (in terms of resource usage) such that resources underutilized by one VM can be utilized by a co-located VM at peak time (Jungsoo, Martino, Ruggiero, & Marcel, 2013; Hu, Liu, Jiang, & Wang, 2015; Meng, et al., 2010). According to this technique, it is assumed that all co-located VMs cannot peak at the same time (Verma et al., 2009). According to Meng et al.

(2010), an efficient technique for VM resource provisioning based on the fact that peaks and valleys in one workload pattern do not necessarily coincide with the others, can be developed. This approach exploits statistical multiplexing where unutilized resources of one VMs can be borrowed by a co-located VMs. Although this method is good, it not suitable in multi-tenant cloud environments and further, it may not be suitable if all the VM peak simultaneously.

Han, Que, Jia and Shu (2016) have proposed a power-aware (PA) algorithm (PA) for determining the most suitable PM to shut-down for energy savings. PA determines the underloaded host for shutdown by dividing hosts power consumption by the number of VMs running in the host. A host which produces the highest value is considered under loaded and thus shut down. The VMs in the underloaded host is moved to another PM which is accomplished by remaining utilization-aware (RUA) algorithm. RUA places a VM in a host if its resource needs are less than the resources remaining in a host. When PA and RUA are combined, simulations experiment results reveal that there is a trade-off between energy consumption due to server utilization and SLA violations. The aim of the used approach is to ensure that all server resources utilization level remains high all the time. Unfortunately, Mar *et al* (2016) assert that power consumption and throughput increase linearly up to a certain point of resource utilization. Aggressive utilization would cause a slight service degradation but a drop in power consumption. Additionally, the degradation caused by aggressive resource utilization causes an increase in execution time, which in turn encroaches into energy saving made from reduced idle energy (Mirabel & Siddiqui, 2015). In this regard, the challenge is to obtain an optimal performance and energy point. Thus, utilizing resources at 100 % may not necessarily lead to energy savings.

Supriya, Rajesh and Anju (2014) proposes a proactive thermo-aware VM placement algorithm, which takes into account current and maximum temperature (threshold temperature) before making a VM placement decision.

The incoming VM is thus placed in a PM, which has the highest difference between the current temperature and the threshold temperature. This is to avoid chances of hardware and software failure and most importantly, to reduce energy used for cooling due to temperature rise caused by VM activation. The same concept can be used to perform thermo-aware VM migration. However, it is hard to predict temperature rise due to VM activation as well as the temperature rise owing to the number of VM already present in the PM. The literature discussed in the techniques of determining the destination of VM migration is summarized in Table 3.2.

Table 3.2: Techniques to determining the destination of a migration literature summary

Technique	**Goal/aim**	**Power saving**	**Strengths**	**Weakness**	**Reference**
Interference-aware VM allocation	Predicting the performance of co-located VMs	By reducing execution time	Minimize interference between virtual machines (VMs)	Uses only benchmark applications as a source of workloads	(Chen et al., 2015)
Multiplexing	Improving server resource utilization via statistical multiplexing	By reducing the number of PMs holding VMs	Uses cloud workloads collected from a production data center	Not suitable in a multi-tenant cloud where VMs resource requirements peak simultaneously	(Meng at al., 2010)
Correlation-aware VM placement	Improving server resource utilization and avoiding server overload	By reducing unnecessary VM migrations	Uses cloud workloads collected from a production data center	Only one resource (CPU) is considered	(Wang, Cheng & Tse, 2016).
Thermal-aware VM placement	Minimizing temperature increase within a data center server as a result of	By reducing the amount of power used by cooling components	It is proactive i.e. it determines the likely temperature increase	It is not trivial to predict temperature rise due to VM activation	(Villebonnet & Georges, 2014; Supriya, Rajesh &

	VM activation.	in the data center	before VM allocation	Anju, 2014)

Additionally, many factors such as current server load (discussed in section 2.3.4.1), temperature (discussed in section 2.3.4.5) and cluster location need to be taken into account before VM placement is decided. For instance, according to Dabbagh et al (2015), cloud data centers are made of clusters located in different locations to exploit reduced electricity costs and thus clusters at regions of lower electricity costs may be selected to host an incoming VM. Once a cluster is selected, the next step is to determine the PM in the cluster that will host an incoming VM. In this case, the authors see VM consolidation problem as a BP optimization problem in which PMs are viewed as bins with different capabilities and VMs as objects with different sizes (resource demands) and the objective is to pack these objects in as few bins as possible. BP problem is known to be NP-hard (Dabbagh et al., 2015). Thus, heuristics such as First Fit (FF), Next Fit (NF), Worst Fit (WF) and Best Fit (BF) can be used to map VM to PM.

Anton and Rajkumar (2011) have examined the problem of VM selection for migration. They have compared Minimization of Migrations (MM) and Highest Potential Growth (HPG) VM selection policies with the Random choice (RC). MM's objective is to select a VM for migration to minimize migration overhead (energy consumed and performance loss) whereas HPG aims at reducing CPU usage to minimize SLA violations. On the other hand, RC selects a VM for migration randomly. The results from their experiments showed that MM outperforms HP and RS.

A VM placement algorithm has two main goals – ensuring QoS and energy savings (Zoha & Shailendra, 2015; Chirag, 2014). Poor VM placement algorithm may lead to triggering of new VM migrations, energy wastage and SLA violations. VM placement is purely an optimization problem. To solve the optimization problem, many researchers have proposed a number of solutions which include

constraint programming (Zhang & Ansari, 2013; Jiankang, Hongbo, & Shiduan, 2015), bin packing (BP) (Zhang & Ansari, 2013; Song et al., 2014; Chaima, 2014; Dabbagh et al., 2015), stochastic integer programming (Speitkamp & Bichler, 2010), genetic algorithms (Zoha & Shailendra, 2015).

Constraint programming has been used in many applications such as planning transportation and timetabling. In the case of VM placement, input data has to be known. In this approach, some constraints have to be put in place so that after VM placement, all the constraints are met. Example of the constraints include; 1) Capacity constrains: for all of the VM dimensions/resources (CPU, memory, hard drive and network), the sum of all resources of the VMs need to be less than or equal to the resources available in the PM, 2) Placement constraint: all VMs must be placed to the available PMs, 3) SLA constraint: VMs are placed in a PM such that agreed between the CSP and cloud user SLA is fulfilled, 4) QoS constraint: VM is placed in a PM such that some QoS constraints are met. In this scenario, an objective function, which is an energy function, has to be minimized. It means VMs are placed in such a way that PM consumes the least amount of energy. Constraint programming is only useful if input data is known in advance such as VM size and the capacity of the total resource of PM.

In Bin packing based VM placement approach, a VM and a PM are considered as a three-dimensional object (dimensions are the resources). VM placement ensures that VMs (items) are placed in PMs (containers) in such a way that the least amount of PMs is used. However, the operation of placing a VM in a PM is slightly different from that of placing items into a container. In the latter, items can be placed side by side and even on top of each other, which is not possible with the former. This has to be captured while formulating the bin packing problem. Bin packing based VM placement gives rise to very important VM placement algorithms, which are First Fit (FF), Worst Fit (WF), Random Fit (RF) and Best Fit (BF) VM placement algorithms (Khan, Paplinski, Khan, Murshed,

& Buyya, 2018). FF algorithm searches through the running machines to host a VM in the first host that can provide the resources required by a VM. If no suitable host is found, a new host is activated. BF picks a PM with the least residual resources while WF picks a PM with the most residual resources. Finally, RF chooses a PM to place a VM at random. If the PM satisfies resource requirement of the VM, the VM is placed in that host, otherwise, another PM is picked at random and the process is repeated.

Genetic algorithm based VM placement uses a class of algorithms which is motivated by natural evolution such as selection and inheritance to create solutions to optimization problems. In this approach, evolution starts with a random population, whose characteristics are modified via genetic operations. In the case of VM allocation, a VM placement approach that uses a genetic algorithm is multi-objective. For instance, it may simultaneously focus on minimizing resource wastage, minimizing energy usage and minimizing of the costs of thermal dissipation (Gohil et al., 2016).

Unlike constraint programming, stochastic integer programming involves optimization problems where data is unknown. For instance, VM placement operation can be taken to be a stochastic integer programming since the resource usage of the VM is unknown, so they can be calculated using their probability distribution.

3.3.2 Energy efficient Virtual machine sizing

A VM size is the measure of computing resources –CPU, Memory and I/O – assigned to a VM (Sareh, 2016). For instance, IaaS cloud VM can be sized to have 1 virtual CPU (vCPU), 1 GB memory, 2000 GB network bandwidth and 25 GB of SSD. VM sizing techniques can be categorized as static or dynamic (Sareh, 2016). Static techniques involve fixing VM sizes and consolidating them in fewer PMs possible or characterizing workloads, then sizing VMs according to application

workload without changing configurations to match later changes of resource demands. Unfortunately, static techniques are not the best options because application workload resource demands change frequently. This situation can be handled by using dynamic VM sizing strategies. In dynamic VM sizing, VM configurations are adjusted at runtime to meet VM applications resource demands. The ultimate objective of dynamic VM sizing is to reduce resource underprovisioning and overprovisioning.

Most IaaS cloud providers require their users to determine the resource requirements of their VMs. For experienced users, this is a trivial task. However, as discussed in section 2.2.4, for inexperienced users, more resources, than required, are assigned often to VMs leading to server underutilization, which is a major cause of resource and energy wastage in the cloud (Patel, et al., 2015). Resource wastage cause energy wastage in the sense that, when resources are underutilized, many physical servers are required than when resources are efficiently utilized. According to an analysis conducted by Dabbagh et al (2015) on Google cluster trace resource usage, 65 % of CPU and 45 % of memory is unused. Thus, new techniques need to be developed to deal with VM sizing amid unpredictable workload changes in VMs.

Underprovisioning, which leads to resource underutilization, can be avoided by using techniques such as resource overcommitment (Cheng, Chai, & Anwar, 2018). This technique involves allocating resources to the VMs than a host PM can afford. For instance, allocating 4 VMs 2GB each of RAM on a PM with 6 GB of RAM. Overcommitment assumes that no VM will utilize all the resources that are allocated to it, thus more VMs can be placed in one PM, hence reducing energy consumed as fewer PMs are used. One downside of resource over-commitment is when the total resource request by VMs exceed what the PM can provide – overload. In overload situations, VM migration is triggered to avoid service degradation thus reducing chances of SLA violation. According to

Dabbagh et al (2015), it is difficult to determine the level of resource overcommitment. To address this, the same author proposes an approach where future aggregate resource demands for the VMs is predicted, which assists in estimating a reasonable overcommitment level. With regard to VM sizing, IaaS cloud providers allow their clients to specify their VM sizes, which is frequently overprovisioned. Unfortunately, the underutilized capacity is still billed (Ganesan, Sarkar, & Narayan, 2012).

Achieving VM sizing may call for other considerations such as known industry standards, which define what levels of resource utilization are acceptable or unacceptable. For instance, the data center Maturity Model (DCMM) is a best practice reference model used for evaluating data center resource usage. According to DCMM, the highest level, otherwise known as *Visionary*, is achieved if the average monthly CPU utilization is above 60% (Xuesong, Barbara, & Monica, 2018). Another reference model is a threshold setting for physical CPU and memory known as VMware Knowledge Base (VMware KB) (VMware, 2015). According to VMware KB, 80% CPU utilization is considered a ceiling and a warning if CPU utilization is 90% for 5 minutes. On the other hand, 85% memory utilization should be considered a ceiling and above 95% for 10 minutes is an alarm state. Arguably, CPU and memory are being considered in these thresholds because their shortages during a short period impact QoS negatively. The thresholds set by such reference models can be used for scaling. For instance, AWS uses threshold-based horizontal scaling where the number of VMs processing workloads are adjusted when certain static thresholds are reached (Tesfatsion, 2018). This method has a high reconfiguration latency. An alternative to this is CPU-Hotplug, whose reconfiguration latency is 1 second and as such, memory can be changed while a VM runs with minimal downtime.

The work by Chen, et al. (2011) in one of the earliest attempts to address the problem of VM sizing. The authors attempt to give a VM its size (resource

demand) based on its contribution to the aggregate resource demands in the host server. An overall outcome is a function of the VMs own resource demand and that of its co-located VMs along the time. Based on this effective size, a VM placement algorithm is developed, which reduces chances of VM migration. Reduced migrations activities reduce energy consumption.

The work by Hadi and Massoud (2016) proposes a mixed technique that achieves VM sizing and placement. This approach takes advantage of hosting dissimilar workloads in the same server, which reduces the number of ON servers, thus reducing energy consumption. To achieve this, the work proposes the creation of copies of a similar VMs in separate servers to reduce its aggregate resource demand on one server. This way, more aggressive consolidation is achieved without performance degradation. Moreover, multiple copies of VMs means that both VMs cannot be overloaded at the same time and this reduces chances of VM migration - an incoming request is processed by the less loaded VM. A secondary benefit for this technique is reliability brought about by having multiple copies of the same VM for processing workloads.

Sareh (2016) has also proposed an architecture for customizing VMs to match workload task characteristics in container-based virtualization. Their approach clusters job tasks based on resource usage and other characteristics such as task length, priority and submission rate, which are then mapped to appropriate VM types (VM sizes). They have used Google Cluster Trace (GCT) to evaluate their technique, which they conclude achieves their goal of reducing energy consumption in data center servers by efficient utilization of resources. Although some research on VM sizing has been done, it is only focused on SaaS (Ganesan, Sarkar, & Narayan, 2012) and Container as a Service (CaaS) (Sareh, 2016). VM sizing in IaaS cloud service remains unaddressed.

Hu et al. (2015) have proposed a new VM resizing strategy with the aim of matching VM capacity with VM load in IaaS cloud. Their approach proposes a

new VM capacity through time division multiplexing. The new VM size parameters are then dynamically adjusted using hypervisor capabilities known as hot-plug. The authors have compared their strategy with migration and they have reported that performance degradation resulting from their approach is much lower and for a short time as compared to VM migration. They have also reported energy savings as compared to VM migration.

Recently, there have been many attempts made by large CSPs to address the problem of VM sizing. As seen earlier, Windows Azure Cloud tailored service ParkMyCloud, Amazon's CloudWatch and Google's cloud VM sizing recommendation service are the current ways of assisting the customers to understand the actual amounts of resources they are using (ParkMyCloud, 2018; Google, 2018; Amazon Web Services, 2018). Unfortunately, all these services are not automatic and are mere recommendation services, which require human action. With the increased complexity of the cloud, such services should be autonomous to reduce error and redundancy. Moreover, CloudWatch, which is the best according to the services it offers is not free – customers have to make an extra payment to use such a service.

3.3.3 Energy efficient virtual machine workload characterization and mixing

Ismael et al. (2013, p.2) define workload as "…*a specific amount of work computed or processed within the data center with defined resource consumption patterns, task submission frequency, task density, task priority and task length*". A system is made of hardware and software but they are not the only factors that affect its performance and energy consumption. The amount of load a system processes also affects performance and energy consumption. Thus, understanding cloud workload is even important than designing new algorithms. A typical system workload may include tasks to be performed, resource demands and task durations. Cloud workloads resource usage data is considered a time series because it consists of observations on how resources are used by applications in

the data center servers or VMs for a period of time. Understanding system workload is the basis for understanding resource utilization, which in turn affects energy consumption. When applications run in the servers, the system is able to record application resource demand in form of logs, which may be analyzed for use in resource planning. Understanding cloud workloads require intensive analysis of resource usage logs. This analysis can show resource demand patterns and resource demands combinations by applications processed in the cloud. This section explains cloud workloads and how understanding cloud workloads can be used to improve cloud resource utilization and reduce energy consumption in the cloud.

3.3.3.1 Workloads in the cloud

The nature of workloads can be described in different ways. According to Nikravesh, Ajila and Lung (2017), a workload can be described according to its pattern - static workload, growing workload, on-and-off workload, periodic workload and unpredictable workload. A static workload has a constant number of user requests per minute. A growing workload's user requests per minute grow rapidly. A periodic workload shows seasonal changes. An on-and-off workload shows user requests that are processed periodically such as in batch processing or in experimentation scenario. Unpredictable workloads are of type periodic but are hard to predict. Due to the growing complexity of the cloud computing paradigm, cloud workloads elicit dynamism. A workload can also be described according to the information that is presented in system logs such as resource usage tasks, task duration, task arrival rate and task volume (Maria, et al., 2016).

There are three major sources of cloud workloads- real workloads, synthetic workload and workloads obtained by using workload generators (Bangari & Rao, 2016; Deborah, Rodrigo, Rajkumar, & Danielo, 2015; Bahga & Madisetti, 2011; Smith & Sommerville, 2011). Real workloads are collected directly from a running

system such as a production data center. Real workloads are the best for investigating cloud workload characteristics. Unfortunately, real workloads are scarce because CSPs are not willing to publicly make them available because of business and confidentiality reasons (Bangari & Rao, 2016). Some of the publicly available real workloads include Clarknet, WorldCup trace 98, Google Cluster Trace (GCT), Grid workload Archive (GWA), Facebook Hadoop workloads, OpenCloud Hadoop workload, Yahoo cluster traces and Eucalyptus IaaS cluster traces (Shen, Beek, & Iosup, 2015; Reiss & Wilkes, 2011; Bangari & Rao, 2016; Delft University of Technology, 2015).

GCT has two versions. The first trace was first described by Reiss and Wilkes (2011) but it is still being investigated by other researchers such as Minet et al. (2018). The second version is an update of the first one and is collected for a period of 29 days. The trace is obfuscated to make sensitive information in the trace disappear. The trace information is organized into tables and each data element has a timestamp expressed in microseconds. The tables are; machine events table, machines attributes table, jobs table, tasks table and task resource usage table, and each table has many related files in CSV format.

1) *Machine events table*: the table records CPU and memory capacity and events as follows: add a machine to a data center, remove a machine from the data center and update the resources available in the data center
2) *Machine attributes table*: it has kernel version, clock speed, IP of the machine being considered.
3) *Jobs events table and task events table*: they describe the events related to a job and task respectively. These are submitted, schedule, evict, fail, finish, kill and update. Tasks events table also contains the priority of the task, the resource request in terms of CPU and memory of a task and placement constraint of each task.

Another provider of publicly available cloud workload logs is Grid Workload Archive (GWA). The main goal of GWA is to provide a platform where researchers and practitioners can share grid workloads (Delf University, 2018). Any person wishing to share their grid workload can do so as long as they are in a database format (SQLite) or text format (CSV). GWA has collected over 13 workloads shown on their website, Materna being the latest. Materna consists of three traces from a distributed data center, namely Materna-trace-1, Materna-trace-2 and Materna-trace-3 with 520 VMs, 527 VMs and 547 VMs respectively. Materna provides service to different organizations featuring different business lines such as government, digital enterprises, IT factory and SAP business consultancy in Germany. The VMs running in the 3 traces are mostly the same and were collected for a period of 3 months and each of the 3 traces contains information representing one month. Because of the difference in the number of VMs in the three traces, it not possible to tell one particular VM in the three traces and hence they cannot be merged. For this reason, one can only work with one trace at a time.

Materna trace is obtained from a VMware ESX environment with 49 Hosts, 69 CPU cores and 6780 GB RAM. The data contains information about 520 VMs in 520 CSV files. The information about each VM contained in each CSV file (see Appendix D) is summarized in Table 3.3.

Table 3.3: Description of information Virtual Machine information contained in GWA-T-13 Materna Trace

Item	Description
Timestamp	This is the epoch timestamp in milliseconds.
CPU cores	This is the number of vCPUs provisioned to the VM.
CPU capacity	This is the vCPU capacity in MHZ. It is given as the product of the number of cores and the speed per core.
CPU usage	CPU capacity that is actually used by workloads in MHz.
CPU usage	CPU capacity that is actually used by workloads in percentage (%).
Memory provisioned	This is the memory capacity for the VM in KB.
Memory usage	this is the actively used memory in KB
Memory usage	This is the actively used memory in percentage (%).
Disk write performance	this is the disk throughput in KB/s.
Disk size	This is the size of the HDD in GB.
Network throughput (received)	This is the network performance in terms of KB/s.
Network throughput (transmitted)	This is the network performance in terms of KB/s.

When GCT and GWA-T-13 Materna compared, GCT has been studied exhaustively by numerous researchers, it is a bit older and represents SaaS and PaaS cloud features. Conversely, GWA-T-13 Materna has not received serious attention, newer and represents a lot of IaaS cloud features.

Because of the scarcity of real cloud workloads, synthetic workloads have become popular (Bangari & Rao, 2016; Costa, Grange, & Courchelle, 2016). Synthetic workloads are generated by algorithms such that their characteristics resemble those of a real workload. Bangari and Rao (2016) have successfully generated synthetic workloads based on the characteristics of GCT using IBM's Statistical Package for the Social Sciences (SPSS) program. Workloads can also be generated using benchmark applications (Smith & Sommerville, 2011). Benchmark

applications are certain software that has been designed to behave in a predefined manner. Workload generators are configurable and can produce a wide range of workloads to meet different scenarios. Currently, many workload generators exist such as Rice University Bidding System (RUBiS) (Rice University , 2017), Phoronix Test Suit (Smith & Sommerville, 2011) and TPC-W (Mar, Lavinia, Orgerie, & Guillaume, 2016).

Workload characterization is very important because it is applied in a number of areas such workload scheduling (Smith & Sommerville, 2011; Mar et al., 2016; Ismael et al., 2013), workload prediction and resource planning (Nikravesh, Ajila, & Lung, 2017; Khan, Yan, Tao, & Anerousis, 2012; Shen, Beek, & Iosup, 2015; Hassan, 2015), synthetic workload generation (Bangari & Rao, 2016; Bahga & Madisetti, 2011; Costa, Grange, & Courchelle, 2016), evaluation of workload failures (Chen X. , 2014; Cano, Aiyar, & Krishnamurthy, 2016) and system performance and security analysis (Maria, et al., 2016).

Quantitative attributes of cloud workloads are important in resource planning because they show resource requirements of applications such as CPU, I/O, network and memory (Maria, et al., 2016). From a resource planning point of view, it is important to understand how workloads behave over time. For instance, resource requirements for some workloads may be stable (show uniform distribution over their execution) while others (such as those for online services) exhibit dynamicity. Another non-functional attribute of workloads is related to SLA constraints. SLA may include performance, security and reliability. Reliability is very important because it is taken seriously particularly when business-critical workloads are deployed in the cloud. To obtain sensible workload attributes, workloads are characterized. Mostly, statistical techniques have been proposed as effective in characterizing cloud workloads.

Shen, Beek & Iosup (2015) have used statistical techniques in characterizing business-critical workloads. The authors base their study on Bitbrain's Grid

Workload Archive Trace 12 (GWA-T-12) dataset. According to the authors, using simple statistical analysis such as min, max, the quartiles, the mean, the median, standard deviation (SDev) and unitless coefficient of variance (CoV), one can achieve considerable workload characterization. For instance, the authors have used CoV to report the level of workload dynamicity. Further, the authors have used the Pearson correlation coefficient to report the correlation between requested resources and used resources, and the correlation among the specific computing resources.

Other statistical techniques that can be used to achieve workload characterization include percentages, percentile, frequency, standard deviation (SDev), cumulative distribution function (CDF) and clustering. For instance, Rasheduzzaman et al. (2014) have used the k-means algorithm to cluster GCT jobs using 11 characteristics mentioned in the work by Reiss & Wilkes (2011) as a feature set. The authors have discovered that the largest clusters have very short time low memory active jobs, while the smallest clusters are very long active jobs, which means that cluster management system does not need to keep inactive jobs in memory. Singh et al. (2010), have used the k-means clustering algorithm to group data center server requests into groups. The resultant groups are then used to generate optimum mix ratio for workloads and estimation of server capacity to minimize SLA violations. Khan, et al., (2012), have applied a clustering technique on a time series of VM resources used to identify groups of VMs, which exhibit correlated workload patterns. Results from using this technique are used to predict variations of workload patterns for resource planning purposes. Similarly, Sareh (2016), has used an extended version of K-means, called X-means, to cluster GCT based on a set of cluster feature set – task length, submission rate, scheduling class, task priority and task resource usage. The goal of the process is to create groups of tasks with similar patterns in terms of their resource usage so that available resources can be allocated efficiently to each unique group. An analysis carried out

by Salam, Karim and Ali (2018) show that clustering is a necessary analysis tool used to gain behavioral knowledge of VMs and cloud users for prediction purpose. This is because it is difficult to predict each type of resource separately for two reasons; 1) VMs have different resources, which makes it difficult to create a prediction technique, 2) Different cloud users may request different amounts of a similar resource. So, it makes sense to cluster VMs and then create prediction models for clusters. Thus, the authors have proposed the use of k-means clustering algorithms for this purpose.

Workload prediction is another form of workload characterization, which can also be said to be a combination of the basic workload techniques discussed earlier (Amiri & Mohammad-Khanli, 2017). According to the dynamic nature of cloud and rapid resource demand, resource provisioning is a problem on the cloud. Resource over-provisioning is costly and wastes resources and energy. On the other hand, resource under-provisioning is detrimental in terms of poor QoS. Thus, fast and accurate models are required to predict future resource demands in the cloud data center for purposes of efficient scheduling. Amiri and Mohammad-Khanli (2017) have surveyed a number of techniques that could be used to create such models. Particularly, the authors have identified Hidden Markov model analysis and Hurst exponent as two methods for predicting future changes in resource demand in the cloud. These methods are applied to historical data, which is found in cloud traces showing applications resource usage. In a cloud computing scenario, the Hurst exponent is used to determine whether the behavior, such as resource usage, of a VM is predictable. Concisely, Hurst is a statistical measure used to classify time series and then reports the level of difficulty in predicting and choosing the best model for a series at hand. Thus, the Hurst exponent cannot be used alone and is only useful when used in conjunction with other techniques. For cloud workloads, which satisfies the Markov property, Anton (2013) used a Markov model to solve the problem of server overload

detection and to predict unknown non-stationary workloads. Markov prediction model can also be used to predict future underutilized/over utilized PM's state to avoid immediate VM migration or to determine the best candidate hosts to host migrated VMs to minimize migration future migrations (Melhem et al, 2018). However, for a Markov model to be suitable, a time series has to posses a Markov property. Moreover, the observed variables need to be discretized to represent states, which need to be predicted and as opposed to real values.

Another set of two techniques, which have been reported to be successful in predicting future application resource usage is Autoregressive Integrated Moving Average (ARIMA) or Artificial Neural Network (ANN). For instance, Ullah, Hassan and Khan (2017) have proposed a data center resource usage prediction that is based on either ARIMA or ANN. Real-time resource usage in a server is monitored at intervals and is used to forecast future resource demands. If the resource usage collected over time follows Gaussian distribution, ARIMA model is used to forecast, otherwise, ANN is used. A target time series is checked for Gaussian distribution using the Jarque-Bera test. The authors report that ANN performs better than ARIMA on a number of accuracy metrics such as Mean Error (ME), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). A similar conclusion has been made in the works by Xue, et al. (2015) and Mozo, Ordozgoiti and Gómez-Canaval (2018).

3.3.3.2 Predicting resource utilization using artificial neural networks

Artificial neural networks (ANN) or simply neural networks (NN) are a class of models used for prediction and classification, which are based on how the human brain neurons are interconnected and learn from experience (Shmueli et al., 2017). ANN mimic how a human expert learns. Although a lot of research has been done on different types of ANNs, one that has received a lot of success in forecasting using multilayer feedforward networks. These networks, such as the one shown in Figure 3.1, has an input layer, hidden layer and output layer. The

input layer is the first layer and it contains nodes, whose number is equivalent to the number of attributes, which are used to predict future values.

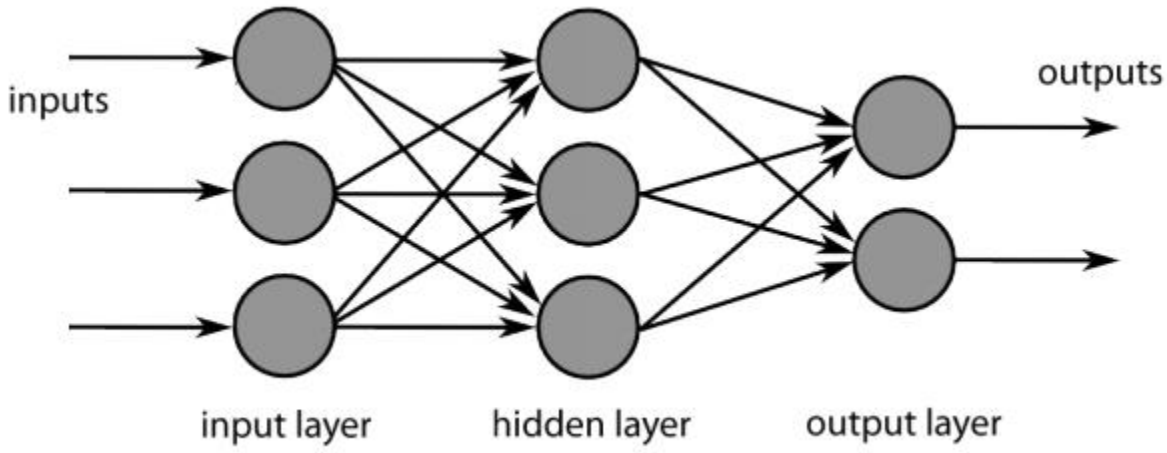


Figure 3.1: Neural Network (Adopted from Abbas, 2015)

The hidden layer serves to improve prediction accuracy and consists of nodes with activation functions. According to Karsoliya (2012), the number of nodes in the hidden layers should be less than twice the number of nodes in the input layer. Therefore, if there are p input nodes then the maximum number of hidden layers is given as $2p$-1. The output layer is used to produce outputs from ANN. The arrows represent some weight and there is a way of mapping an input of a node to output using some function, which is called an activation function.

The most commonly used sigmoid activation function is the logistic function, which is shown in equation 3.3. For some input X, it produces some output $g(X)$ that is between 0 and 1; $0 < g(X) < 1$

$$g(X) = \frac{1}{1 + e^{-X}} \qquad 3.3$$

ANN models are used in forecasting of time series. Although ARIMA models can be used in time series forecasting as well, they call for many assumptions. For instance, ARIMA assumes that the time series is stationary, that the errors in the forecast follow a particular model and that probability distribution of observations is Gaussian (Shmueli et al., 2017). If these assumptions are not true, then ARIMA forecast can be very inaccurate. Luckily, ANN does not require as many assumptions as ARIMA and thus can be applied to non-linear functions without much analysis of the underlying assumptions. However, if such underlying assumptions can be discovered, ANN models can be more accurate.

Just like other machine learning models, an ANN model is trained. A dataset is split into a training set and test set (Bankole, 2013). During training, the dataset is used to update the network parameters such as the weights. Validation set may be used during the training period to monitor the process of training to detect scenarios such as over-fitting. Starting values for the model weight are near zero random values (Bankole, 2013). The model thus starts as a linear model and grows non-linear as the weights increase. Use of exact zeros as starting weight values leads to zero derivatives and thus the model never moves. On the other hand, starting with very large values leads to a poor model. A model has to be trained for some period of time thus a stopping rule has to be set to avoid over-fitting. There are three terms that are very important in ANN which are epoch, learning rate and momentum.

Epoch is when the entire training dataset is passed through the network forward and backward once i.e. when ANN is trained on every training data item ones. A single epoch is not enough because it leads to under-fitting. As the number of epoch increase, the curves moves to the optimum and to over-fitting and thus the model has to note the optimum. Sometimes a bias value is required to avoid over-fitting. Learning rate is the rate at which weights change. Weights are the values which are used to adjust the model. As an ANN uses gradient descent

optimization algorithm to minimize error function to reach a global minimum, momentum is the amount of influence from the previous iteration in the present. Momentum should be chosen carefully because it can cause a model to get stuck in a local minimum.

Neural network techniques have been successful in predicting cloud resource usage as reported by various researchers (Prevost et al., 2011; Frey et al., 2015; Duggan et al., 2017; Xu et al., 2016; Lu, Panneerselvam, Liu, & Wu, 2016; Cao, et al., 2018). For instance, Frey et al. (2015), have proposed an ANN model for determining the amount and type of storage that will be fit to host workload in order to reduce SLA violation based on the predicted performance requirements of workloads. The approach has been compared with the threshold based approach to determining the amount of storage and results show that this approach performs better in terms of SLA violation reduction.

The work by Duggan et al. (2017) is motivated by the fact that the problem facing cloud computing is the low accuracy of predicting future resource usage. The authors have proposed an ANN model to predict multiple host CPU utilization values into the future using historical host CPU utilization. The authors have reported that using recurrent neural networks, which have the ability to retain information, have shown great potential in forecasting time series data.

As reported by Ullah, Hassan and Khan (2017), ANN can perform prediction on multiple time series as well as predict variations in workload patterns via clustering. This means that in the context of resource usage prediction, one can predict multiple time series by using individual historical observations and the correlations among the time series.

3.3.3.3 Energy efficient workload consolidation characterization and mixing

Workload consolidation that does not consider workload profiles before workload placement can lead to performance loss and energy wastage due to

interference caused by applications competing for the same physical resources (Sareh et al., 2015). If VMs with similar profiles are co-located, there is increased workload interference. As a result, workload tasks run longer and more energy is consumed. Therefore, it is prudent to co-locate a mixture of workloads that have characteristics, which are consolidation friendly.

Smith and Sommerville (2011) have investigated the effect of different workloads on server power consumption in a private cloud using PTS workload benchmarks. The authors found that placing many VMs, which dominate one computing resource in the same PM, has a detrimental effect in terms of performance and power consumption. Experiments conducted by the authors reveal that it is wise to pair VMs, which do not consume a large amount of a similar resource. For instance, pairing a CPU intensive VM with a disk intensive VM in the same PM is advisable. From this point of view, workloads can be categorized into homogeneous and heterogeneous. Homogeneous workloads exist when a set of applications behave in a similar manner such as consuming only a single resource in a server. On the other hand, heterogeneous workload exists when a set of applications behave differently such as consuming different resources in a server. A heterogeneous workload can also mean a workload that has a mixture of long and short runnings tasks. To explain the benefits of co-locating dissimilar workloads, Dhiman (2011) has conducted experiments by running homogeneous and heterogeneous workloads in a server in turns using eon and mcf workload benchmarks. He has concluded that homogeneous workloads run for a longer time than heterogeneous thus consuming more energy.

Liu and Cho (2014) have characterized workloads on GCT. The frequency and pattern of jobs and task-level workload behavior, and how the overall cluster resources are utilized is studied. The success rate of jobs and tasks are studied i.e. successful tasks and jobs and those that eventually fail or get killed. The authors have concluded that if the cloud resource scheduler is offered hints about the

nature and periodicity of the submitted jobs, it may specialize the resource management decisions in order to save energy and reduce performance variations of important jobs.

Using Windows Live Messenger and Windows Azure workload traces, Guenter, Jain, and Williams (2011) describes an energy-aware server provisioning strategy that predicts near future resource demands via load patterns analysis, auto-correlations and cluster utilization. Their objective is to minimize unmet resource demands while reducing energy usage and cost of hosting clusters. Ensuring that unmet resource demands are reduced, minimizes the number killed tasks, which in turn reduces energy consumption because each task is performed once. This strategy has been tested on three different workloads traces and results reported shows that energy saving on a greedy algorithm are about 96%.

Mar et al (2016) have investigated the effect of workload profiles on power consumption. The authors used the TPC-W workload generator tool and varied client behavior using *browsing* and *ordering* profiles. They observed that power consumption is greater under ordering than under browsing while both have the same throughput. Thus, it is wise to determine an optimal workload mix (ordering and browsing), which delivers energy savings.

Xia, Lan and Zhao (2014) have proposed a technique for energy saving in IaaS cloud via migration. They use K-means to cluster workloads, which is the basis of detecting PM overload and under load that triggers VM migration. Using this characterization, their technique is able to determine resource demands in real-time on the different clusters so that VM scheduling is done efficiently hence reducing energy consumption. Energy is reduced by ensuring that server utilization is averagely high thus tasks run in less time. GCT cloud workload trace is used to evaluate this technique. Maintaining a high resource utilization as a basis for achieving energy savings through future resource demand forecasting and provisioning has also been studied by Patel et al (2015) and Singh et al (2010).

Sareh et al. (2015) have proposed a new architecture, which allocates groups of tasks to customized VMs based on task characteristics in container-based clouds. This mapping is based on actual task resource usage patterns obtained from an analysis of real usage trace logs instead of the resources requested by cloud users. The authors have used the K-means algorithm for clustering. The main components of the architecture include a task classifier, task mapper, VM type definer, power monitor, host controller and VM controller. The task classifier receives a stream of tasks and clusters them based on task length, submission rate, resource usage and priority. Suitable VM types of the classes of tasks are identified and they are sent to the task mapper. The task mapper instantiates suitable VMs provided by the VM type *definer* for the different classes of tasks. The power monitor estimates the power usage of the cloud data center servers based on the resource utilization on the data center servers. The host controller runs in each data center servers and it periodically monitors resource usage in the servers. It also sends levels of resource utilization to the power monitor. Finally, the VM controller runs in each VM to monitor resource usage of tasks as compared to provisioned resources. The goal the proposed architecture is to map groups tasks to customized VM sizes, which exactly meet the required resources. By so doing, energy consumption is reduced by efficiently allocating resources to tasks. By the fact that the VM controller runs in the VM, this architecture cannot be used in IaaS cloud model because CSP is not allowed to execute any application inside a user VM.

Sondhi, Gupta and Vivek (2016) have used k-means to group cloudlets (task to be mapped to a VM) using instruction size, execution deadline and cost paid by the customer as the clustering feature set. The Euclidean distance is computed using the three clustering feature set. As such, the priority of an incoming cloudlet is determined by the three parameters. The authors have reported that when their technique is compared to base techniques (existing work), there is an improvement

in power consumption, total turnaround time, wait time, processing time and processing cost. Moreover, the work by Sheenam & Navtej (2016) has reported an improvement in performance while using a similar approach.

Al-Dulaimy et al. (2016) present a technique for consolidation where jobs to be processed are classified based on their resource usage. Thus, any incoming job's resource usage can be determined based on the group to which it belongs. The groups of jobs are determined by clustering. Moreover, it is easy to map an incoming job to the right VM size because its resource consumption is already known. In addition, clustering ensures that VMs running similar jobs are not placed in similar physical servers. The objective of this work is to better utilize the involved physical server resources, which minimizes energy consumption. Although the authors have not disclosed the algorithm used to cluster the jobs, clustering has been used.

Hadi and Massoud (2016), have proposed an algorithm based on dynamic programming that takes advantage of scheduling dissimilar workloads in the same server. This approach is meant to reduce server energy consumption by consolidating workload in fewer servers. The authors observe that co-locating highly correlated workloads lead to VM migrations, which decreases performance. Lowly correlated workload ensures that servers are well parked and idle power well utilized. The approach used in this work is creating copies of VMs in different physical servers and then distributing the incoming requests to these VMs. This reduces the chance of running a similar workload in the same server. The most applicable cloud service model for this approach is SaaS.

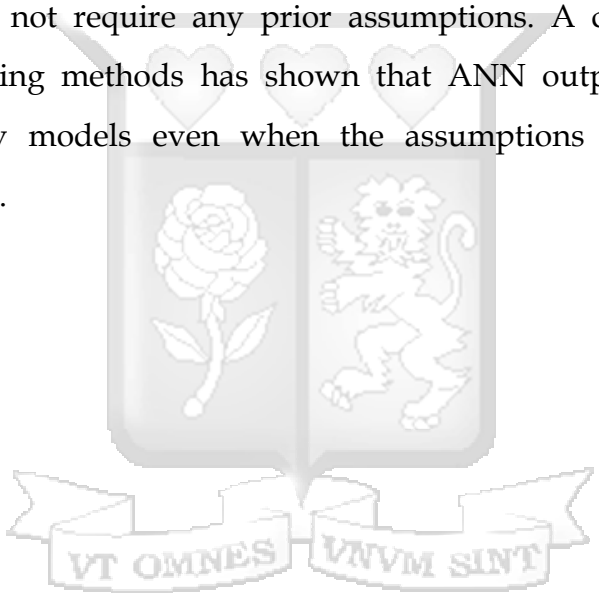
The work by Chen, et al. (2015) a technique for predicting the level of interference and impact on the performance of co-located VMs. With this approach, workloads are mapped to a VM, which will lead to low interference with co-tenant VMs. The authors have used micro benchmark applications to

generate workloads – *sysbench* for CPU intensive workloads and *fio* for disk-intensive workloads (Joel, 2017). Although the author does not mention how energy consumption is reduced, the work by Ismael et al. (2013) has shown that an increase in interference among co-residence VM decreases energy efficiency.

3.4 Summary

In this chapter, the existing techniques used to ensure energy efficient in cloud data center servers have been reviewed. Various research work that focuses on the same subject has also been reviewed. It can be concluded that the main cause of energy wastage in cloud data center servers is resource wastage caused by low server utilization. This problem can be addressed by workload consolidation. However, workload consolidation that does not consider workload profiles lead to performance loss and energy wastage due to interference caused by applications sharing the same physical resources. Workload profiles can only be achieved via workload characterization and so far, a number of statistical techniques for characterizing cloud workloads have been reviewed. Moreover, the review shows that the techniques for workload characterization need to be combined to achieve a complete understanding of cloud workloads. Most of the work that has been reviewed lack the analysis of real workload traces to inform consolidation decisions and even the ones that have done so, such the work by Sareh et al. (2016) and Hadi and Massoud (2016), do not target IaaS public clouds and all have used similar workload traces, GCT. Other approaches such as DVFS energy savings hardware capabilities have failed. DVFS fails because it is designed for the CPU since the other computing resources' power ranges are narrow. On the other hand, energy savings hardware capabilities can only be ensured by the hardware manufacturer and hence no improvements can be possible during workload execution.

The problem of resource overprovisioning can be addressed by VM sizing. Unfortunately, due to varying nature of resource demands by cloud workloads, VMs cannot be allocated fixed resources. This calls for techniques for forecasting resources usage in the data center. Different forecasting techniques, which include Hurst exponent, Hidden Markov models, ARIMA and ANN have been reviewed. The Hurst exponent, Hidden Markov models and ARIMA have various weaknesses. For instance, Hurst exponent cannot be used without other techniques whereas Hidden Markov models and ARIMA assume certain properties of the time series. ANN remains to be a suitable forecasting method because it does not require any prior assumptions. A comparative study on various forecasting methods has shown that ANN outperforms ARIMA and Hidden Markov models even when the assumptions in question are true (Almqvist, 2019).



Chapter 4

Performance and Power Profiles of Virtual Machine Workloads

4.1 Introduction

The purpose of this chapter is to investigate: 1) the impact of workload type on power consumption and performance of server hosting VMs and 2) the impact of the level of server consolidation on power consumption and performance of server hosting VMs. Workload type means if a cloud workload is homogeneous or heterogeneous. The goal of this investigation is to find out the existence of consolidation friendly workloads and those that are not good for consolidation.

As mentioned in section 2.2.4, in IaaS public cloud, cloud users are allocated a VM, which is mapped to a PM and users are free to use any type of OS and run any type of applications at will without any control from the CSP. Thus, VMs running applications with similar (homogeneous) or different (heterogeneous) profiles can be mapped to the same or different PM. Experiments were conducted using PTS suites as the source of application workloads.

4.2 Phoronix Test Suites (PTS) as a Source of Workloads

According to Smith & Sommerville (2011), Vasudevan, et al. (2010), Martin & Marangozova-Martin (2018) and Phoronix Test Suite (2018) PTS is the most compressive open-source benchmarking tool. PTS has a collection of test profiles with different but known behaviors. PTS has been used successfully in testing computer performance, hardware validation and as a source of cloud workload. The test suites in PTS are developed to run in a number of operating systems such as Microsoft Windows, MacOSX, Linux, Solaris and Berkeley Software Design (BSD). Today, PTS has over 450 benchmark micro-programs which target hardware performance and stability such as disk, processor, GPU, memory, network and

overall computing system. When a benchmark is run, PTS uses sensors to read different variables such as CPU power consumption, system power consumption, CPU frequency, memory usage, CPU temperature and I/O speed (Phoronix Test Suite, 2018a). Version 8.2.0 of PTS includes a server known as Phoromatic, which automates the process of test orchestration to any number of client systems and can be used to schedule tests that require to run at particular times.

When PTS test suites are used as a source cloud workload, they are tagged depending on the computing resource that is targeted. For instance, to test an aspect of system CPU such as performance, a test suite known as 7zip is used (Smith & Sommerville, 2011). The following is a description of four selected test suites, which were used in this study as a source workload and is summarized in *Table 4.1*.

Table 4.1: Summary of selected PTS tests

Targeted resource	Select PTS test
CPU	7zip
Memory	gzip
Network	Loopback TCP network
Hard disk	aio-stress

7zip

7zip is a file compression application using 7-zip compression with an observed overhead on CPU and a bit of memory. Hence it is labeled as a CPU benchmark or CPU intensive workload. 7zip is designed to compress a large file during which the CPU demand remains constantly high (mostly at 100%) as shown in Figure *4.1*. The current version of 7zip is 1.7.0 as was used in this study.

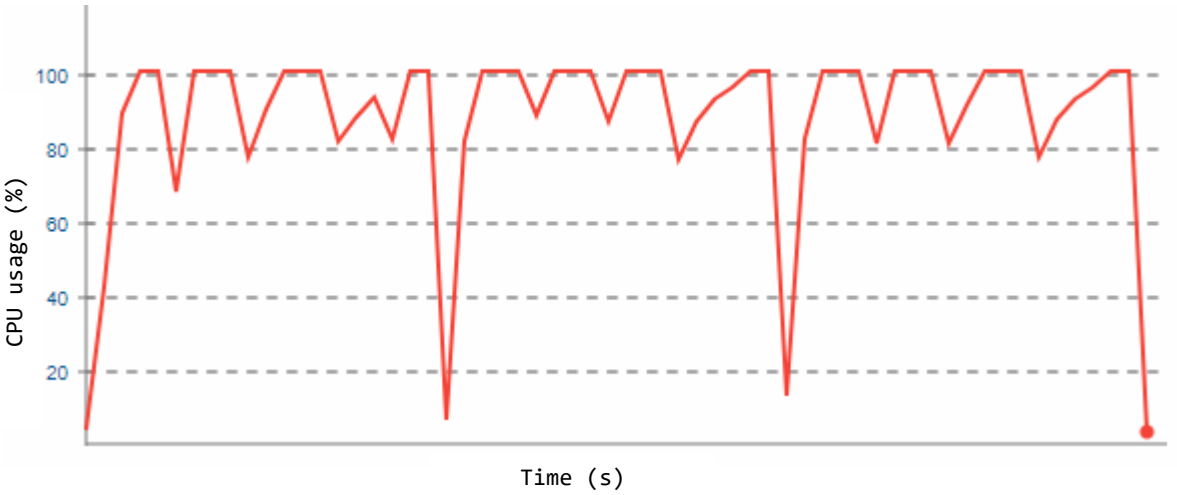


Figure 4.1: 7zip CPU usage in percentage over time

gzip

This is a memory PTS benchmark, which measures the time it takes to compress two copies of version 4.13 of Linux kernel source tree using gzip compression. This application is observed to have a high overhead of memory, as shown in Figure 4.2, thus it is labeled as a memory-intensive workload. The size of the Linux kernel source tree is about 2GB. 7zip and gzip performs almost a similar task but with an observed difference in the computing resources demand, which dominates their operation.

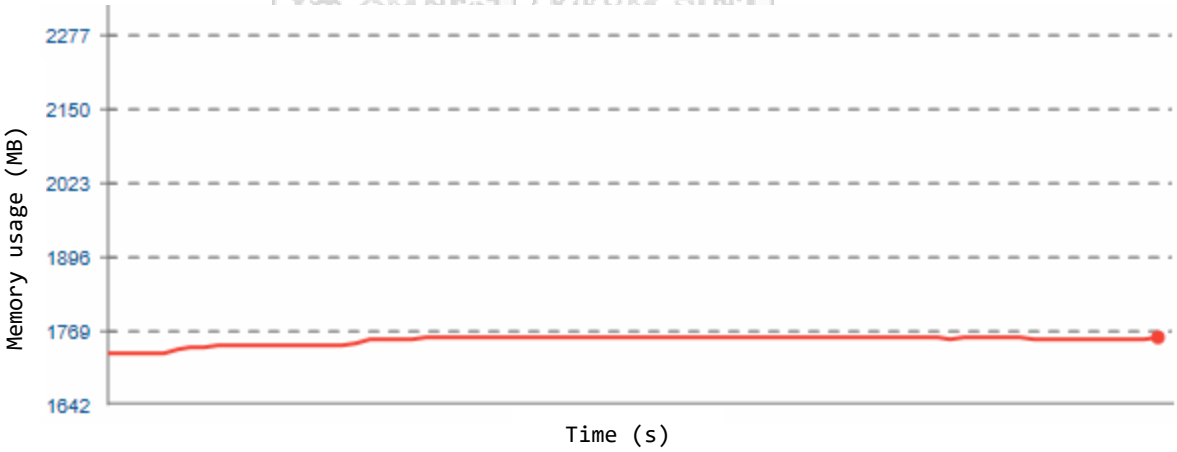


Figure 4.2: gzip memory usage in megabytes over time

aio-stress

This is an asynchronous I/O benchmark created by Software und System-Entwicklung (SUSE) (software and systems development). The current profile (version 1.1.1), which was used in this study, uses a single thread and constantly reads and writes a 2048 MB test file and a 64KB block size from and to the hard disk. For this reason, the system subcomponent that is being tested is the hard disk.

Loopback TCP network

This benchmark puts an activity on the loopback network adapter to measure its performance. It simulates network traffic and thus is it labeled as a network-intensive workload.

4.3 Experiment design and setup

The four selected PTS test suites shown in Table 4.1 were used as the source of application workloads. A server, whose properties are summarized in Table 4.2, was set up to host eight VMs. Kernel-based Virtual Machine (KVM) hypervisor was used to virtualize the server hardware and Ubuntu server OS was used in both the host server and VMs. Phoromatic server was also set up in a Personal Computer (PC) to be used to automate the process of orchestrating PTS workloads, which were already installed on the server. The Phoromatic server operations were accessed via a web client as shown in *Figure 4.3*. Phoromatic server is set-up in a separate server to avoid its processes from interfering with the behavior the PTS test suite applications. To investigate the impact of workload type on power consumption and performance of server hosting VMs, workload type was varied by either using homogeneous workloads or heterogeneous workloads. Homogenous workload consisted of four VMs executing a similar PTS suite and a heterogeneous workload consisted of four VMs each executing a different PTS suite. For homogeneous workload four VMs executed 7zip, gzip, aio-stress and loopback network in turns.

Each execution was repeated three times to verify results obtained. During each execution, average power consumption over time of the server and the performance of execution of the PTS test suites were recorded. Power consumed was measured in Watts (W) whereas the performance of execution of 7zip, aio-stress, loopback network and gzip were measured Million Instructions per Second (MIPS), Megabytes per second (MB/s), seconds (s) and seconds (s) respectively. 7zip performance was measured in MIPS because the suite was targeting the CPU and thus MIPS was used measure the performance of the CPU. Likewise, gzip performance was measured in seconds because it was targeting the server memory and thus it was determining how fast the server memory could compress two Linux kernel source tree using gzip compression. Aio-stress performance was measured in Megabytes per Second because it was targeting the hard disk to measure the speed of writing and reading of a 2048 MB size file from/to a hard disk. Loopback network performance was in Megabytes per Second because it was targeting the network interface to measure the speed the network interface could achieve on simulated network traffic. Power measurement was accomplished using Linux powerstat component on Running Average Power Limit (RAPL) interface (explained in section 2.3.1, also see Appendix C). Performance measurements were obtained from Phoromatic server via the web interface. As control mechanism, each PTS test suite was executed independently in one VM. Power consumption of the host server and performance of each type of workloads were tabulated.

To investigate the impact of the level of server consolidation on server power consumption and performance of the server hosting VMs, the level of server consolidation was varied by increasing or decreasing the number VM running and executing PTS test suites in the server. In this case, one type of workload (homogeneous) was used but was performed in two different PTS test suites. For instance, one VM running 7zip test suite was executed and power consumption of the server and performance of the VM measured. The number of VMs executing

7zip was increased one by one to a maximum of eight VMs and each time, power consumption of the server and performance of the VM was measured. The same procedure was repeated using aio-stress instead of 7zip. The results were tabulated and plotted.

Table 4.2: A summary of characteristics of Physical Server (host) and Virtual Machines

Item	**Item details**
Physical server (host)	Processor: Intel Core i7-4790 @ 4.00GHz, 8 cores 8192 KB cache size. Operating system: Ubuntu 18.04, 4.15.0-36-generic (x86_64) kernel version Memory size: 12288MB Hard drive (HDD): 1000GB Western Digital WD10EZEX-60M.
Virtual machine (guest)	Processor: 1 vCPU @ 3.59GHz Operating system: Ubuntu 18.04, 4.15.0-36-generic (x86_64) kernel version Memory size: 1280MB Disk size: 43GB Hypervisor/system layer: KVM
Number of VMs (guests)	8

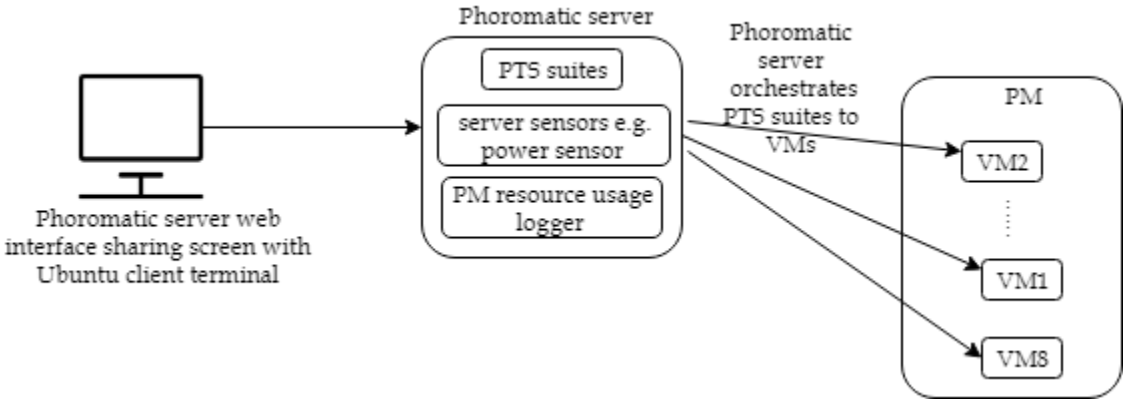


Figure 4.3: Architecture for executing PTS test suits using Phoromatic server

4.4 Experiment results and discussion

Table 4.3 summarizes the performance and power consumption when homogenous and heterogeneous workloads are compared. It was observed that the performance when executing a particular workload is highest when a single VM was executed in the physical server than when multiple VMs were executed. For instance, running 1 VM of 7zip attained performance of 3737 MIPS (green cell) as compared to 3099 MIPS (yellow cell) when multiple instances (4 VMs) of 7zip was executed. On the other hand, multiple VM execution's performance was better with heterogeneous VMs as compared to homogeneous VMs. For example, CPU performance using 7zip achieved 3099 MIPS (yellow cell) with heterogeneous workloads as compared to 2985 MIPS (red cell) with homogenous workloads. When a single VM is executed in a PM, there is no competition to use the physical resources and hypervisor capacity. However, when more VMs are consolidated in a single server, more VM instances compete for the physical resources as well as the hypervisor capacity resulting in interference. The interference causes tasks running longer due to low throughput and thus consume more energy – energy is a product of power and time and given in equation 2.2. This interference was severe in homogenous workloads because only one resource (a particular

resource) was used, which created a hotspot of activity while the rest of the resources remained idle.

Table 4.3 also shows that when 4 homogeneous VMs were executed with 7zip, 46.48 watts (blue cell) of power was consumed. When this was repeated with 4 heterogeneous VMs (running 4 different suites), power consumption reduced to 31.74 watts (orange cell). The reason for this decrease is attributable to the elimination of a hotspot of activity, which dominated one resource and was spread to other computing resources. The high amount of power observed in homogenous workloads can also be attributable to the the idle power of unused resources, which goes to waste.

Table 4.4 shows the average power consumption and performance for both 7zip and aio-stress when they were separately executed at different levels of consolidation. It also captures performance per watt when executing 7zip. These results were plotted as shown in *Figure 4.4* and *Figure 4.5*. It can be observed that power consumption while executing 7zip increased with the level of consolidation increased (*Figure 4.4 a*). An increase in the level of consolidation had an unnoticeable impact on the power consumption caused by executing aio-stress. On the other hand, the performance of executing aio-stress was observed to drop at a faster rate as compared to 7zip when the level of consolidation increased (*Figure 4.4 b*). These results show that the consolidation limiting factor for CPU (due to 7zip) is power consumption and that of the hard disk (due to aio-stress) is performance. Although power consumption increased with level of consolidation, it can be observed that the power performance (performance per watt) increased (*Figure 4.5 b*). This is an advantage of workload consolidation from a power savings perspective.

The noticed performance degradation caused by aggressive consolidation (rapidly increasing level of consolidation) is as a result of interference from the co-

resident VMs. *Table 4.5* shows the measure of interference when 7zip and aio-stress weree executed at different levels of consolidation. Interference was represented by a value called Total interference index (TII), which was computed according to equation 4.1.

$$TII = \sum_{i=1}^{k} \frac{P' - P}{P'} \qquad 4.1$$

where k is the number of VMs co-located in the same physical server, P' is the performance of 1 VM when it runs alone in the physical server and P is the performance of the VM when it runs with other VMs in the physical server. The data in Table 4.5 were plotted in Figure 4.6 and it shows that an increase in the number of VMs (i.e. level of consolidation) increased performance interference. It is also observed from the plot that the rate of increase in interference in aio-stress higher than that of 7zip. This observation further supports the argument that the limiting factor for disk intensive workloads is performance.

Table 4.3: Summary of Performance and Power profiles for Homogenous and Heterogeneous VMs.

	Average Performance				Average Power consumption (W)			
	CPU for 7zip (MIPS)	**Memory for gzip (s)**	**Hard disk/ IO for aio-stress (MB/s)**	**Network for network loopback (s)**	**CPU for 7zip**	**Memory for gzip**	**Hard disk/ IO for aio-stress**	**Network for network loopback**
Executing each of the 4 workloads in 1 VM running in physical server separately	3737	68	57.57	15.62	23.39	23.99	11.71	15.36
Executing heterogeneous workloads in 4 VMs	3099	160	28.99	16.36	31.74			
Executing homogenous workloads in 4 VMs	2985	501	10.35	16.98	46.48	19.00	10.36	15.39
Idle server (99.8% idle)	____	_____	_____	_____	4.55			

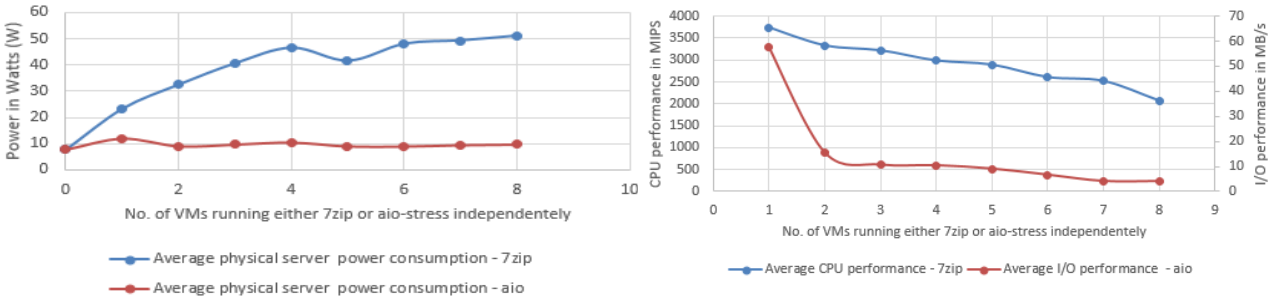
Table 4.4: Performance, power consumption and performance per watt when processing 7zip and aio-stress at different levels of server consolidation

No. of VMS/level of consolidation	Average power consumption in Watts (W) - 7zip	Average power consumption in Watts (W) - aio-stress	VM/CPU performance in MIPS - 7zip	VM or I/O performance in MB/s - aio-stress	Performance per Watt in MIPS/W - 7zip
0	7.88	7.88	_____	____	0
1	23.39	11.71	3737	57.57	159.77
2	32.55	9.07	3329	15.38	204.55
3	40.61	9.645	3215.67	10.57	237.55
4	46.48	10.36	2985	10.35	256.88
5	41.58	9.008	2890	8.99	347.52
6	48.014	9	2606.1667	6.55	325.68
7	49.26	9.45	2518.571	4	357.9
8	51.085	9.67	2071.25	3.98	324.36

*0 number of means that the server was not executing any workload in any VM- the server was idle

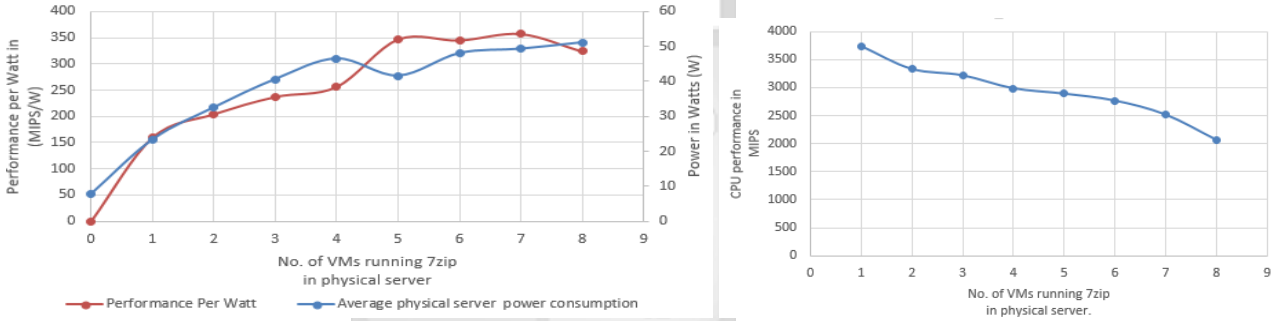
Table 4.5: Total Interference Index (TII) for 7zip and aio-stress at different levels of consolidation

No. of VMS/level of consolidation	Total interference index (*TII*) - 7zip	Total interference index (*TII*) – aio-stress
0	____	____
1	0	0
2	0.22	1.46
3	0.42	2.45
4	0.8	3.28
5	1.14	4.29
6	1.81	5.32
7	2.28	6.51
8	3.09	7.71



a) Power consumption b) Performance

Figure 4.4: A comparison between CPU and I/O's performance and power consumption when processing 7zip and aio-stress respectively at different levels of consolidation



a) Power and power performance b) CPU performance

Figure 4.5: Effect of increasing the number of VMs on performance, power consumption and power performance

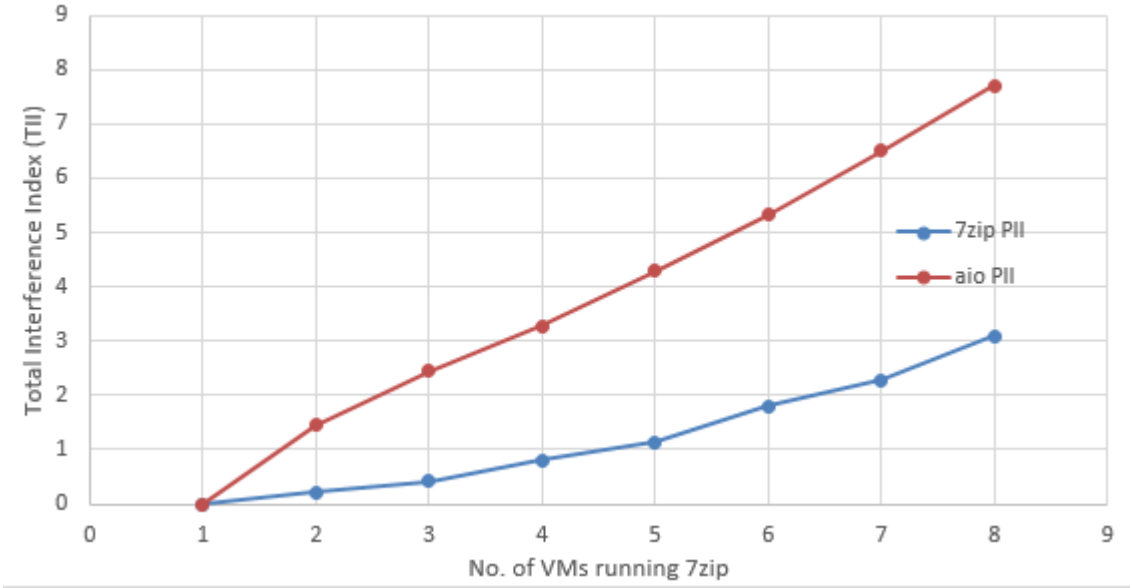


Figure 4.6: The Impact of Performance Interference at Different Levels of consolidation on Homogenous Workloads

4.5 Conclusion

From the results of the conducted experiment, a number of conclusions were drawn. First, application workloads have different characteristics of power and performance profiles, which is caused by the types of resources they use. For instance, 7zip used the CPU and as a result, its execution consumed more power as compared to aio-stress, which used the hard disk. On the other hand, I/O (hard disk) intensive workloads are highly affected by interference as compared to CPU intensive workloads. Efficient power usage can be achieved via consolidation because of improved power performance (performance per watt). Moreover, use of PPW metric is important because it shows how efficiently the energy that goes into processing workload processing is used. However, due to the overhead of the virtual layer, the increased competition of consolidated workloads on the shared resources cause interference leading to performance loss. This is more pronounced in the homogeneous workloads as compared to heterogeneous workloads. Thus, it is advisable to co-locate heterogeneous workloads because it ensures that

processing activities are distributed to all computing resources and hence runs quickly thus saving energy. The implication of this observation is that VM allocation algorithms need to take into account the characteristics (resource usage) of VM workloads. Although consolidation is advantageous from an energy efficiency point of view, it can lead to serious performance loss for certain types of workloads as observed in the case of aio-stress. In this regard, it can be concluded that workload consolidation limiting factor for CPU intensive workloads and disk intensive workloads is power consumption and performance respectively.

Chapter 5

Virtual Machine (VM) Allocation Algorithm

5.1 Introduction

In the previous chapter, the benefits of co-locating heterogeneous workloads over homogeneous workloads were identified. Indeed, there is improved performance and energy savings achieved when dissimilar workloads are co-located. This benefit is not always achieved when consolidating workloads in IaaS public clouds as CSPs does not have control of the types of applications executed by users in VMs. This means that VMs with similar profiles can find themselves in the same PM. This chapter describes the design and evaluation of a VM allocation algorithm, which ensures that VMs with similar profiles do not get co-located during placement. The main contributions of this chapter are:

- An architecture for VM resource usage profiling, VM clustering and VM allocation with the aim of reducing energy consumption in IaaS cloud data center was proposed.
- An approach for clustering real VM resource usage logs for VM allocation was proposed. By extension, this section provides early insights towards an understanding of Grid Workload Archive Trace 13 (GWA-T-13) Materna cloud dataset.

The next sections explain the target cloud service model and the components of the proposed architecture.

5.2 Target cloud model

In this section, the cloud model is large scale public IaaS owned by an organization, that provides cloud services to individuals and small organizations. In this cloud model, users request a VM to be created by selecting from predefined virtual machine types (sizes). The VMs are then created and placed on available

PMs by the hypervisor/ Virtual Machine Monitor (VMM) as shown in *Figure 5.1*. Users run applications on their specific VMs. The user has control of the VM and can configure and execute any type of application. From the CSP point of view, applications are a black box host in a VM. However, in public clouds, users do not have access to VMM, only the CSPs does (Jiaqing, Nipun, & Willy, 2010). To understand the resource usage of the application running in the cloud, the CSP has to profile VMs via the VMM layer. It is assumed that the CSP has in place an effective method of monitoring VM resource usage. The dataset that was used showed the resources actively used by the VM, which made it sufficient for this work (Delf University, 2018) and was explained earlier in chapter 3.3.3.1.

5.3 System architecture

The proposed architecture is shown in *Figure 5.2* and its components are explained in this section. This architecture is based on previous architecture, which was successfully used by other researchers. For instance, a proactive dynamic VM consolidation framework used in the work by Salam, Karim an Ali (2018) shows the presence of the following components;- a monitoring tool, which monitors resource usage by applications, database to hold monitored resource usage by VMs, a clustering component, which puts VMs into groups for easier analysis and a host profiler, which check the state of resources in the PM. An architecture proposed by Sareh (2016) for mapping applications tasks to the right VM advocates for high granularity i.e. components, which perform specific roles. Because the architecture's goal is to minimize power consumption, a component for estimating the power consumption of the cloud data center based on the resource utilization of the available hosts is required. The power monitor must reside inside the PM (host). Moreover, the architecture should be such a way that the proposed solution processes do not introduce overhead into the resources being managed (Tesfatsion, 2018). In this regard, a black-box approach needs to be used where a monitoring system gets its statistics outside a functional unit.

Therefore, the proposed system should run in a separate host from the ones whose resources are managed. In our case, the IaaS VM *reconsolidator* runs in a separate system away from the cloud resources being managed.

The first part was called *IaaS VM reconsolidator* because the process of consolidating a VM can happen again if a VM profile changes due to changes of the applications executed in the VM.

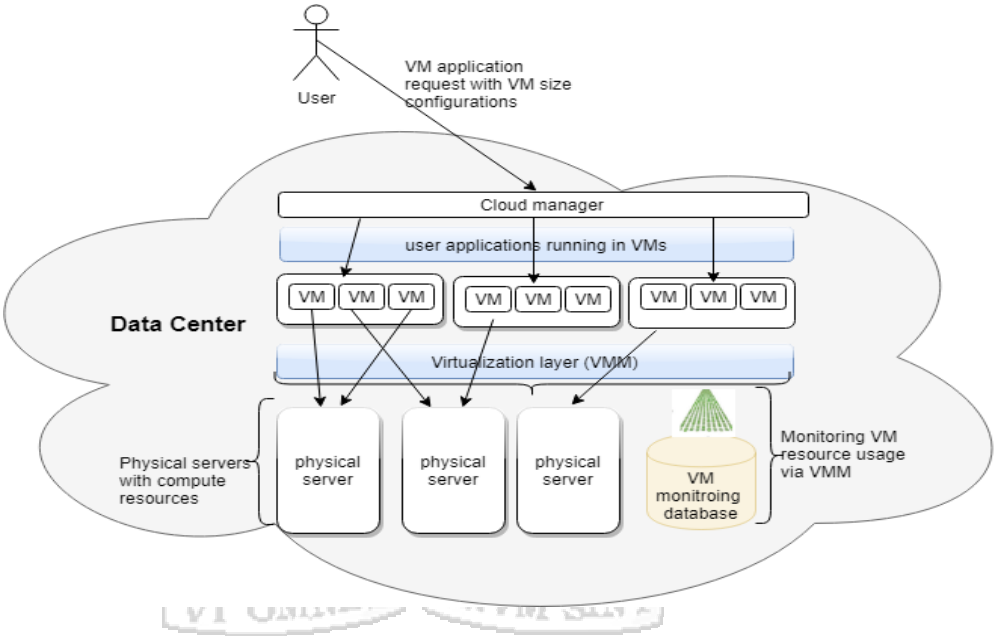


Figure 5.1: Overview of Cloud Model

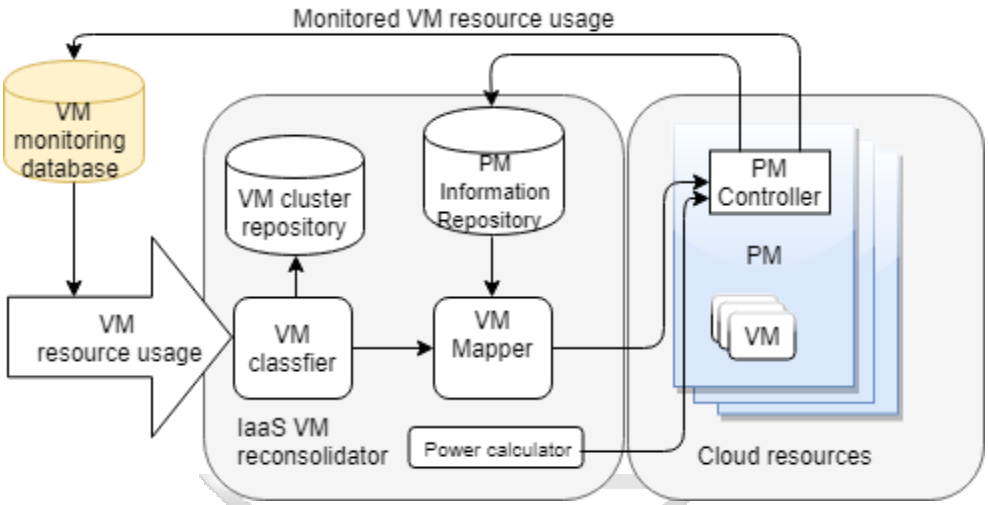


Figure 5.2: Proposed system architecture

1) *VM Classifier*: this component is used to classify VMs based on their historical resource usage. It is trained using historical data harvested from VMs. It receives VM resource usage from the VM monitoring database and then classifies it based on CPU usage and memory usage. The complete process of clustering is discussed in section 5.4. After a VM has been classified, the classification results are stored in a VM cluster repository and also forwarded to the VM mapper. The classification frequency is determined by the CSP based on how frequent VM profiles change in the physical servers of the data center.
2) *VM Mapper:* this component receives classification results from the VM classifier and determines the new host for the classified VM. From the host list, all hosts are fetched and are referred to as candidate hosts (*candidateHostList, see algorithm 5.1*), which have enough resources to accommodate the classified VM. The candidate hosts are then sorted in order of increasing similarity of VMs in running hosts with the classified

VM. Similarity Index, *I*, of a host machine with the classified VM is computed as shown in Equation 5.1.

$$I_i = \frac{k_i}{n_i} \qquad 5.1$$

where *k* is the number of VMs in the i_{th} host machine that shares a group with classified VM and *n* is the total number of VMs in the i_{th} host machine. The first host in the sorted candidate host becomes the new host. The complete operation of VM mapper is shown in Algorithm 5.1.

Algorithm 5.1: VM Mapper Operation

Input: *hostList*, *classfiedVm*
Output: *newVmHost*
1. **for** *host* in *hostList* **do**
2. **if** *host* has enough resources to accommodate *classfiedVm* **then**
3. *candidateHostList*.add(*host*)
4. **end if**
5. **end for**
6. **for** *host* in *candidateHostList* **do**
7 *I = k/n* //compute I for each host in *candidateHostList*
8. **end for**
9. sort *candidateHostList* using *I*, ASC.
10. *newHost = candidateHostList.get(0)*
11. **return** *newHost*

3) *PM Controller*: this component runs in the Physical Machine (PM). It periodically checks resource utilization in the PM caused by VM utilization and sends it to the *PM information repository*. Since the IaaS CSP cannot install monitors in the rented VMs, the PM controller also monitors resource usage of the VMs via virtualization layer and stores it in the VM monitoring database.
4) *PM Information Repository*: this component stores information regarding data center hosts. For instance, it is the source of host list input in Algorithm 4.1.

5) *Power calculator*: this is a simple component that estimates the power consumed by all active hosts at any given time during the execution of an application. Total power is calculated according to equation 5.2**.**

$$P_{total} = \sum_{i=1}^{k} ((P_i' - P_i) * \left(\frac{n_i}{100}\right) + P_i \qquad 5.2$$

where k is the number of active hosts at any time, P' is the peak power consumption of the i^{th} host, P is the host idle power and n is the percentage CPU utilization of the host. Equation 5.2 is derived from equation 2.3. The former estimates power consumption of one host whereas the former estimates power consumption for a number of hosts.

5.4 Virtual machine clustering using k-means

K-means clustering is a type of unsupervised learning, which is used when there exists unlabeled data which data without groups or categories. The goal of k-means is to create k groups with data items which share a similarity in certain features. K means has to be determined before the algorithms run. In this study, the purpose of clustering is to group a number of given VMs such that VMs that exhibit similar behavior in terms of resource usage do not get co-located. k (which equals 4) was determined using the elbow method. Grid Workload Archive Trace 13 (GWA-T-13) Materna (explained in section 3.3.3.1) cloud dataset has been used as a source cloud workload in this procedure. This dataset has a pool of 520 VMs showing data on how resources were being used in CSV format (see appendix D). Basic k-means has been used as a clustering algorithm. For each VM, the following has been used as the k-means clustering feature set:

1) *VM CPU usage as CPU*: the average CPU actually used by the VM for the entire profiling period

2) *VM memory usage as RAM*: the average memory actually used by the VM for the entire profiling period

Recalculation of point closeness to a centroid is computed using Euclidian distance according to equation 5.3.

$$d = \sqrt{(CPU_1 - CPU_2)^2 + (RAM_1 - RAM_2)^2} \qquad 5.3$$

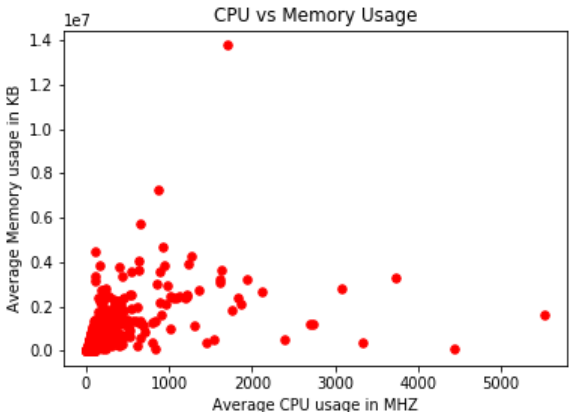
where (CPU_1, RAM_1) and (CPU_2, RAM_2) are points in the Cartesian plane having CPU on the x-axis and RAM on the y-axis. Scikit-learn, a python machine learning open source library, which includes k-means clustering was used (Scikit-learn, 2018). The input to the k-means algorithm, *k*, was determined using the elbow method. The average CPU and memory usage for each of the 520 VMs was computed and used to cluster the VMs. The next section explains the results of VM clustering.

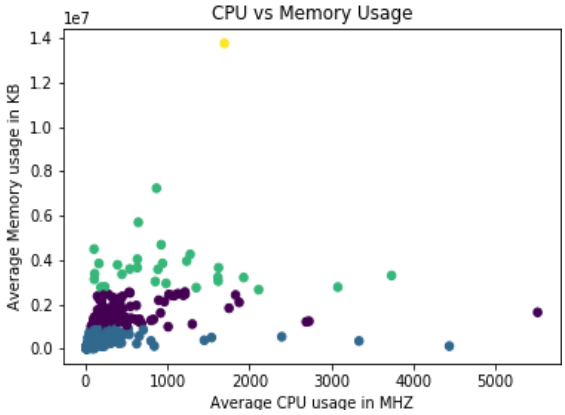
5.5 Virtual machine clustering results

As mentioned in section 5.4, the elbow method used to determine *k* as an input to k-means revealed that the optimal value for *k* was four (4). The population of VMs in each cluster is summarized in Table 5.1 below. Figure 5.3 (a) and (b) shows scatter plots before and after k-means clustering. From Table 5.1 or Figure 5.3 (b), it can be observed that Large VM had only one member and was considered an outlier. The description of the four resultant VM groups are as follows:

Table 5.1: Population of VMs in each resulting cluster

Cluster VM type	**Number of VMs**	**% population of each VM type**
Extra small VMs	394	75.77 %
Small VMs	96	18.46 %
Medium VMs	29	5.58 %
Large VM	1	0.19 %
Total	**520**	**100**





a) Scatter plot before clustering b) scatter plot after clustering

Figure 5.3: Appearance of a scatter plot before and after k-means. Notice the yellow point VM, which we have treated as an outlier. In (b), the yellow dots represent large VMs, green dots represent medium VMs, purple dots represent small VMs and blue dots represent extra small VMs.

1) *Extra small VMs:* this group has a population for 394 VMs out of a total of 520 VMs, which represents 75.77%. Most of the VMs in this group have generally used a very small amount of both memory and CPU except some, around 3, whose CPU demand was high.
2) *Small VMs*: this group has a population of 96 VMs of a total of 520 VMs, which represents 18.46 %. The amount of memory used by these VMs is low but is greater than that of extra small VMs. Generally, the amount of CPU used in this group seems to have not changed significantly when compared with the extra small VMs group.
3) *Medium VMs:* this group has a population of 29 VMs out of a total of 520 VMs, which represents 5.58 %. The amount of memory used by VMs in this group is higher than VMs in extra small and small VMs group. Similarly, the amount of CPU used in this group seems to have not changed significantly.

4) *Large VM:* this group has only 1 VM out of a total of 520 VMs, which represents 0.19 %. This one VM is considered as an outlier because of its position as compared to the other groups. This VM has a high memory consumption with a moderate CPU consumption.

From the foregoing, it can be concluded that memory usage was very important in putting the VMs in their respective groups. Moreover, CPU usage was generally low.

5.6 Evaluation of virtual machine allocation algorithm

This section presents the evaluation procedure of the VM allocation algorithm together with a discussion of the evaluation results.

5.6.1 Virtual machine allocation algorithm evaluation procedure

In order to apply the proposed architecture using real workload traces, GWA-T-13 Materna dataset, which contains information about VMs hosted in a data center that supports business-critical workloads, was utilized.

Further, the algorithm is evaluated by simulating it using a cloud simulator known as CloudSim Plus (Manoel et al., 2017), which is a fork of CloudSim (Rodrigo et al., 2011). CloudSim Plus' data center configurations, which is shown in Table 5.2 was used for evaluation. The data characteristics shown in the table were provided by the publishers of the GWA-T-13 Materna dataset. This cloud simulator is written in Java language. CloudSim and CloudSim Plus are almost similar cloud simulators except that CloudSim Plus has been re-engineered to remove code duplication and to ensure code compliance to software engineering standards. Besides, CloudSim Plus has more features, such as event listeners, than CloudSim and is easier to use. CloudSim Plus components are a *data center*, a *Host*, a *Broker*, a *VM* and a *Cloudlet*. A data center represents the core infrastructure, which is hardware and software. A data center holds hosts, which are computing nodes with a specific set of computing resources (CPU core, memory, hard disk

and network bandwidth). With virtualization, a host holds VMs, which are rented by customers to run user applications. A cloudlet in CloudSim Plus is synonymous to user applications or workload, which require computing resources. A broker is used to submit user applications for processing. CloudSim Plus provides a base or abstract classes, which can be extended and interfaces, which can be implemented to change the way resources are managed in a cloud computing environment. For instance, *VmAllocationPolicy* is an abstract class, one can use to implement own algorithm for deciding on the host that runs a particular VM. This class has one member method called *findHostForVM* which has to be modified to create a new algorithm. In this study, this abstract class was extended in the process of implementing the proposed VM allocation algorithm. The data center, hosts, VMs and cloudlets configurations are based on workload traces used in this evaluation. In addition, the data center configuration shown in Table 5.2 has been provided by the publishers of the workload traces (Delft University of Technology, 2015).

The steps of creating a data center and running a simulation in CloudSim Plus is as follows;-

Step 1:

A simulation was Initialize by creating CloudSim object

Step 2:

A data center was created. CloudSim Plus allows the creation of many data centers. At least one data center is required to run a simulation. Creation of a data center included the creation of hosts (PMs). For each host created, specify the resources available in that host, resource provisioner in percentage and the idle power in percentage of peak power. Add all hosts to a list and then add the hosts to the data center. Finally, specify the VM allocation policy for the data center and other data center characteristics such as hypervisor, operating system and computing resources costs. In this case, for example, various VM allocation

policies such as FF, BF, WF and the proposed VM allocation algorithms were tested.

Step 3:

A broker was created for the data center.

Step 4:

Virtual machines were Created and added to a List. Each VM is created needs to have its size specified i.e. resources assigned to that VM. In this case, 520 VMs from GWA-T-13 Materna dataset was created.

Step 5:

Cloudlets were created and added to a List. For each cloudlet, specify the resource demands. In this case, we have ensured that each cloudlet runs in a specific VM so as to satisfy our cloud model. Thus, a cloudlet is bound to a VM to enable the broker to submit the cloudlet to a particular VM.

Step 6:

The simulation was started. For each simulation, one has to specify the VM allocation policy that needs to be tested – this is explained in step 2. Finally, for each simulation enable results printing.

Table 5.2: CloudSim Plus data center configurations used for evaluation

No. of hosts	49
No. of VMs	520
No. of CPUs	69 (454 cores)
Memory size (in GB)	6780
Hypervisor	VMware ESX
No. of cores allocated per VM	Varying (1,2,4,6 and 8)
Memory size allocated per host (in GB)	Varying (2,4,8 and 16)
Host static power	60 % of host peak power

In the evaluation, the designed VM allocation algorithm was compared with the existing VM allocation policies, which were retained in CloudSim Plus from CloudSim (Sajitha & Subhajini, 2018). For instance, the algorithm is compared with well-known First Fit (FF), Worst Fit (WF) and Best Fit (BF) VM allocation algorithms in terms of execution time and energy consumption (Khan et al., 2018). FF algorithm searches through the active PMs to place a VM in the first host that can provide the resources required by a VM. If no suitable host is found, a new one is activated. BF picks a PM with the least residual resources while WF picks a PM with the most residual resources. This evaluation followed an IaaS cloud multi-tenant cloud model, which was explained in section 4. For this reason, each application runs in a specific VM. To ensure correctness, all the algorithms were tested on a similar data center with similar configurations such as power monitoring intervals, same power model, same VM scheduling intervals. Additionally, all algorithms did not attempt any optimization such as VM migration. The performance metrics adopted were total power consumption and execution time. A summary of the evaluation process is shown in Figure 5.4.

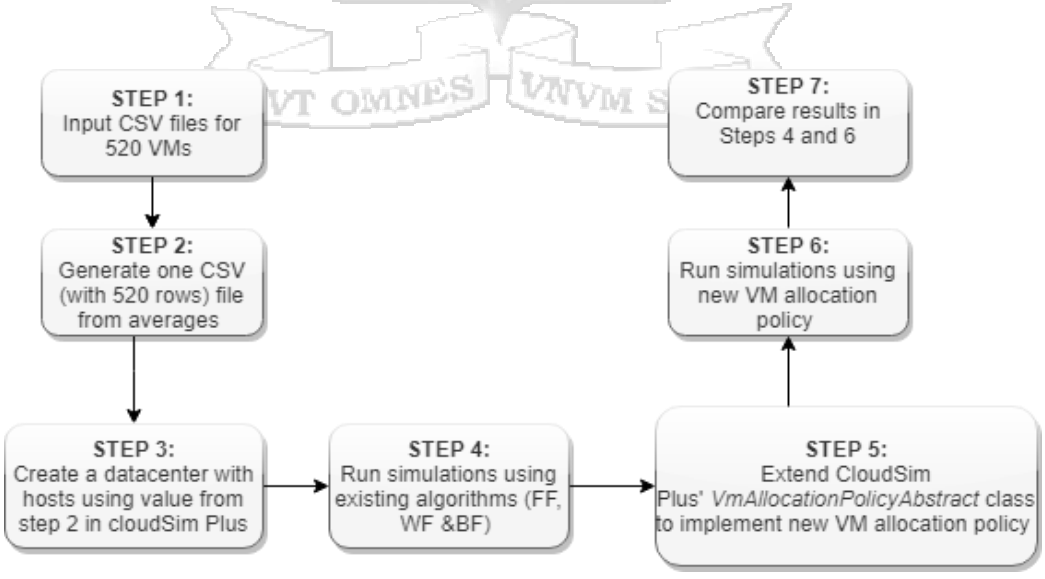
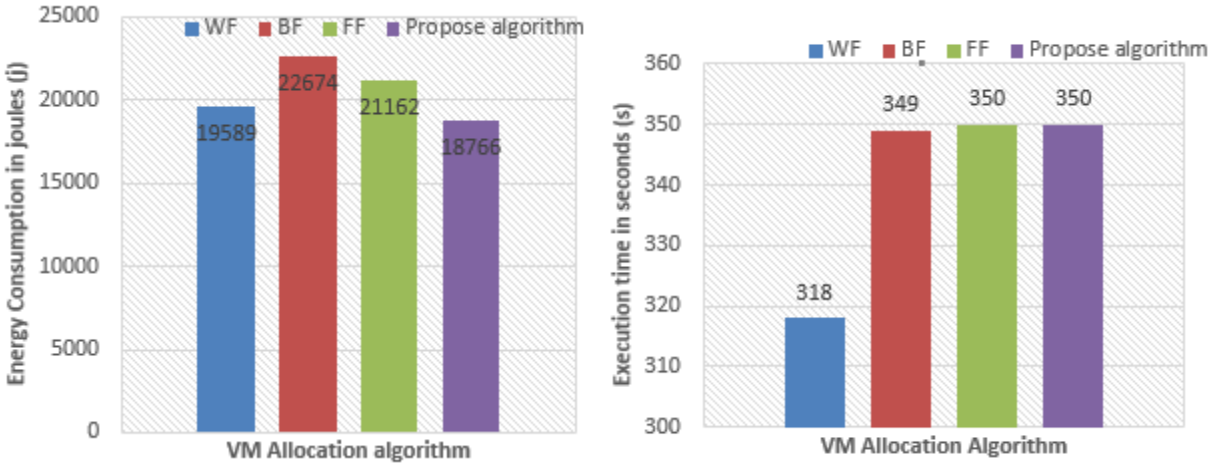


Figure 5.4: Process of evaluation of the proposed algorithm

5.6.2 Virtual machine allocation algorithm evaluation results and discussion

The results of the evaluation are shown in Figure 5.5 (a) and (b). The figures show the total amount of energy consumed by all the 46 hosts and the total time of execution respectively when executing the dataset workload using different algorithms.

The evaluation compared the proposed VM allocation algorithm with WF, BF and FF. The first noticeable observation is that the proposed algorithm consumed the least amount of energy, which is 18767 joules, as compared to the other algorithms. For instance, BF consumes the highest amount of energy, 22673 joules while WF and FF consumed 19589 and 21162 joules respectively. FFSI is efficient because it places a VM in a host with least similar VMs in terms of resource requirements, which reduces the interference caused by resource contention, thus making good use of idle power of all the involved computing resources. The removal of a hot-spot of activity from one resource ensures that idle power is used efficiently, which spares the dynamic power of the computing resources. It is also observed that WF beats FF and BF in terms of energy usage. This is because WF chooses a host with the most residual resources, hence it does not lead to more aggressive utilization and in which case host's idle power is utilized well. VMs using WF allocation policy have plenty of resources and it is the reason why it uses the least time for processing as shown in Figure 5.5 (b)



a) Total amount of energy consumed by all the 49 hosts for different VM allocation algorithm

b) Total execution time for the different VM allocation algorithm

Figure 5.5: The evaluation results of the proposed algorithm. The performance (time taken) and energy consumption as compared with that of FF, BF and WF

The proposed VM allocation algorithm co-locate dissimilar VMs in the same servers, which means that at any given time, all the server components are processing workloads. The proposed algorithm did not achieve low execution time thus its lower energy consumption is attributable to low power usage over time as compared to the other algorithms. The reason for this is because idle power for the servers was well utilized due to the heterogeneity of the co-located VMs. In the absence of homogeneous workloads, which puts pressure on the same component, heterogeneous workloads distribute the workload to all components to utilize all the components idle power. None of these algorithms attempted to improve performance by use of migration. Therefore, these results demonstrate that it is possible to reduce the execution time of the proposed algorithm through VM migration. These results also show that there exists a potential of achieving energy savings if VM allocation policies would consider VM resource usage before scheduling. In this study, we have considered only two resources, CPU and memory, but it can be extended to cover other resources such as network and hard drive to achieve even more energy savings.

5.7 Conclusion

In this chapter, an architecture for VM resource usage profiling, VM clustering and VM allocation were proposed. The central goal of this chapter was to design a VM allocation algorithm, which ensures that VMs that exhibit similar profiles are not co-located so that energy saving is achieved while preserving performance. A clustering approach using K-means was used to put a pool of VMs into groups based on their resource usage.

Clustering results from an analysis of real cloud workloads showed that indeed there exists VMs, which vary or are similar in terms of resources usage. From an earlier conclusion in Chapter 4, if similar VMs are co-located, there is a negative impact on performance and energy savings. Therefore, there is an opportunity of using these results to implement VM allocation policies, which consider resource usage profiles of VMs. From this view, a VM allocation algorithm was implemented and evaluated on a CloudSim Plus simulator. The simulated experiment results indicated that scheduling heterogeneous workloads is beneficial from an energy consumption perspective. Thus in the design of a new VM allocation algorithm, one needs to take into account the characteristics of workloads running in the VMs.

Chapter 6

Virtual Machine Sizing and Virtual Machine Resources Usage Prediction Algorithm

6.1 Introduction

The VM allocation algorithm designed in Chapter 5 only ensures that heterogeneous VMs are co-located. It does not solve the problem of resource wastage caused by resource overprovisioning. In this chapter, the problem of resource overprovisioning and under-provisioning due to data center resource demand dynamism is addressed. This way, the problem of resource wastage, poor QoS and the growing complexity of cloud computing infrastructure management is addressed. Resource management involves resource allocation, resource provisioning and resource monitoring (David & Anbuselvi, 2016). To reduce the work of data center administrators, it calls for use of autonomic systems, which are self-managing. Autonomous systems are needed because of the highly varying and unpredictable nature of cloud workloads, which run on highly heterogeneous infrastructure. Autonomous Resource Management System (ARMS) for performance and energy management must have certain characteristics, which include;

1) *Resource usage monitoring*: ARMS need to monitor resource usage of entities such as VMs and then use to predict future resource usage.
2) *Adaptability*: ARMS need to adjust resources according to the demands of running workloads.
3) *Low overhead*: ARMS operations should not incur significant time and resource usage overheads. In this regard, a black-box approach needs to be used where a monitoring system gets its statistics outside a functional unit.

4) *Timely detection of changes and accurate:* ARMS should detect changes in resource demands in a timely manner as well as avoid over- and under-provisioning of resources.

The main contributions of this section are:

1. A proposed algorithm for VM sizing based on historical resource usage for improving resource utilization with the aim of reducing both data center energy consumption and cloud user monthly bills.
2. A proposed algorithm for predicting future VM resource usage for VM auto-scaling via on-time resource provisioning.

6.2 Materials and methods

To understand the nature of cloud workloads' resource consumption, various methods were used to characterize our workload before proposing an architecture. The cloud workload used for VM sizing was GWA-T-13 Materna, which was explained in section 3.3.3.3. For some reasons, such as errors in the computer program used to put collected data into CSV or lack of system monitoring, some data may be missing. In the workload data that was used, some column's data was missing and were indicated as zero (see Appendix D column C). For instance, for some VM CSV, the column showing CPU capacity provisioned in MHz had zeros (refer to Appendix D column C). This was obviously an omission since the VM had executed workloads and its CPU usage had consistent values. Using the reported CPU usage percentage, this column's values, $CPU_{provisioned}$, were calculated according to **equation 6.1.**

$$CPU_{provisioned} = \frac{CPU_{used}}{CPU_{\%}} * 100 \qquad 6.1$$

where CPU_{used} is the value in 'CPU usage' column and $CPU_{\%}$ is the value in 'CPU usage [%]' column.

To visually inspect resource usage patterns in the workload used, resource usage over time was plotted. To remove higher frequencies that may be in the time series, a 150-point Simple Moving Averages (SMA), SMA_n, filter was applied to remove the higher noise frequency for easier visualization. The values for 150-point simple moving averages were obtained according to equation **6.2**.

$$SMA_n = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad 6.2$$

where n is the number observations in the moving average and y_i is the value at period i. A comparison between resources (CPU and memory) used by the VMs and resources allocated to the VMs was accomplished by plotting graphs on selected VMs. This was carried out to establish if VMs were allocated the resources actually needed or there was overprovisioning. Average resource usage peak to mean ration was also computed to establish resources usage dynamicity. To establish if percentile resource consumption could be used to size the VMs, a comparison between resources used and 90th percentile resource was plotted. To determine the presence of correlation between the values of observation of resource usage time series, autocorrelation was used for lag 5 minutes (lag 1) to 2 hours. Because fixed size VM sizing may be inappropriate when resources demand from VMs exceed the allocated resources, predicting future resources is important for provisioning. Therefore, the distribution of the variables in the resources usage (CPU and memory) time series was tested using skewness and Jarque-Bera (JB). ARIMA would be used for predicting future resources usage if the variables were normally distributed, otherwise, ANN would be used. To complement resource prediction, we investigated if some VMs with peak resource usage could borrow resources from other co-resident VMs, which were not experiencing peaks. For each of the 520 VMs, all peak points, their corresponding

timestamps and if there are peaks in VMs, which occurred simultaneously were assessed. Algorithm 6.1 shows how this was achieved. In this case, a peak point of any VM is defined as any point whose value is higher than a 90th percentile value for that particular VM. The results from the analysis of the workloads will be used to propose a VM sizing architecture. The goal of the proposed architecture was to resize VMs so that the resources assigned to them match their resources demand. To evaluate the performance of VM sizing technique, a data center was setup in CloudSim Plus to compare the energy consumed before and after VM sizing when processing GWA-T-13 Materna workload across four different VM allocation algorithms. The VM allocation algorithms used were FF, BF, WF and the VM allocation algorithm that was designed in Chapter 5.

Algorithm 6.1: Finding the number of VM peaks that occur simultaneously

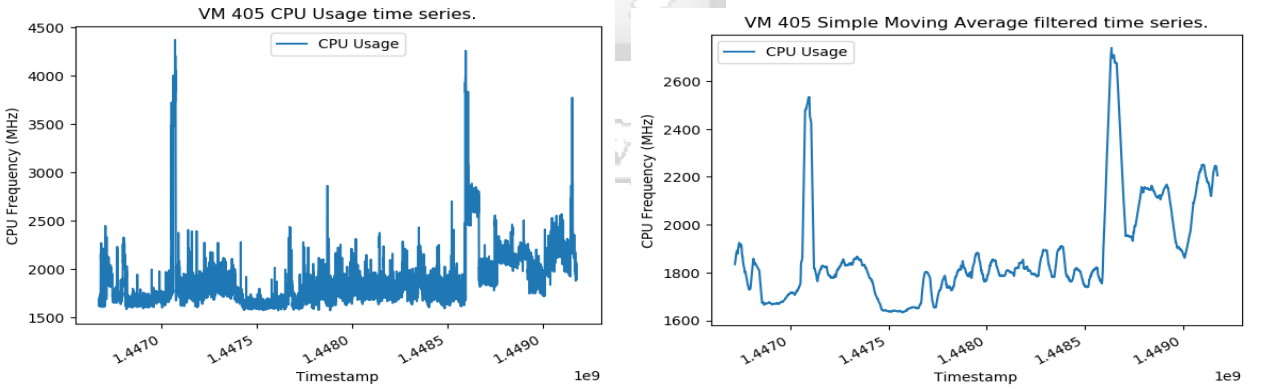
```
1   Input: # of VMs n, a set of time series for each VM Y_t = {y_t1, y_t2, ....y_tn-1, y_tn}
2   Output: uniquePeakCountList // a list of the frequency of 'peak points'
3   foreach y_t in Y_t do
4     percentileValue = get90thPercentileFor y_t ()
5     foreach value in y_t do
6       if value is greater than percentileValue then
7         get timestamp for this value
8         add timestamp to list timestampList
9       end if
10    end for
11  end for
12  peakUniqueList = timestampList.getUnique //get all unique timestamps from list
    /*find the number of times each unique appears in timestampList */
13  foreach unique in peakUniqueList do
14    count = timestampList.count(unique) //the frequency of unique in timestampList
      //add count to uniquePeakCountList
15    uniquePeakCountList.add(count)
16  end for
17  return uniquePeakCountList
```

6.3 Workload analysis results and discussion

Figure 6.1 **(a)** shows a raw plot of CPU usage of VM 405 over a period of time. Figure 6.1 **(b)** shows the resultant 150-point SMA filtered time series. It was

observed that CPU usage over time was highly dynamic and showed big differences between peak usage and minimum usage. This means that it is not trivial to accurately allocate the required resources to the VMs. To address this, public clouds users tend to overprovision resources or size VMs using peak usage. This practice of sizing VMs using peak resources for highly dynamic workloads results in resources wastage.

It was also noticed that the resource usage of almost all VMs was very low. For instance, Figure 6.2 shows memory provisioned and memory that was used to process workloads for VM 01. It is clear that the VM's memory usage was extremely low as compared to provisioned memory. Summarily, the average CPU and memory usage for the 520 VMs was 4.5% and 8.3% respectively. In fact, the percentage of VMs with CPU and memory utilization below 20% was 96.3% and 90.6% respectively. The average highest VM CPU and memory usage stood at 69.3 % and 82.2 % respectively.



a) CPU usage time series for VM 405

b) CPU usage Simple Moving Average (window=150) filter series for VM 405

Figure 6.1: Visualization of VM 405 CPU usage for the entire period. a) The first plot shows all points. b) The second plot shows CPU usage with a 150-point simple moving average

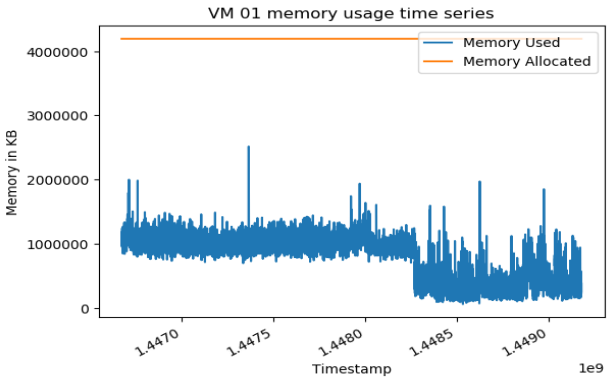


Figure 6.2: A plot of memory allocated, and memory used for VM 01

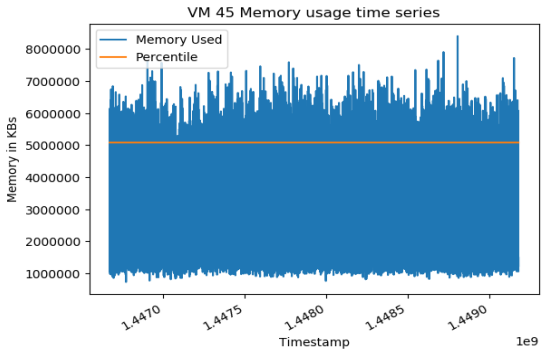


Figure 6.3: A plot of memory usage showing 90th percentile for VM 45

The CPU cores provisioned to the VMs are 1, 2, 4, 6 and 8. 78.8% of the VMs had either 1 or 2 cores and the rest had 4, 6 or 8 cores. Average resource usage peak to mean ration, which was used to show dynamicity was 16:1 for CPU and 10:1 for memory. Figure 6.3 to Figure 6.7 shows CPU and memory usage's 90th percentile for various VMs (01, 45, 467 and 492). These results indicate that fixing VM resource allocation to VMs using 90th percentile will not be sufficient because there are sometimes when resources usage is extremely high than its 90th percentile. The results from the analysis of peak points for each of the 520 VMs, showed that, indeed, there were peak points occurring simultaneously but such instances were very low. This result creates an opportunity of using statistical multiplexing (Meng, et al., 2010). Statistical multiplexing is exploited where unutilized resources of one VM can be borrowed by co-located VMs. Out of the 520 VMs (with over 4.3 million data points or observations), a memory peak at timestamp 1.447967e+09 happened simultaneously only in 25% of the VMs and this is the highest among peak points identified for memory. For CPU, a peek at timestamp 1.449137e+09 happened simultaneously only in 24% of the VMs and it was also the highest among peak points identified for CPU. Other peaks for CPU and memory occur simultaneously in less than 24% and 25% of the VMs respectively. These results also mean that resource over-commitment does not

create problems with these workloads as resource demands did not peak at the same time more frequently.

Figure 6.6 shows a frequency distribution plot of memory usage for VM 172. It was observed that the distribution has a positive skewness (skew value = 1.9021426). Skew values for VM 45 CPU usage, VM 172 CPU usage, and VM 45 memory usage were 5.2309817, 2.670302 and 1.6542248 respectively. Additionally, Table 6.1 shows Jarque-Bera (JB) results for memory usage from VM 172. The computed *p* value was lower than the alpha value (0.05), thus it was concluded that the variable in consideration was not normally distributed. JB values for memory and CPU usage were calculated on VM 01, VM45 and VM 492 and results showed that the variables were not normally distributed. In this study, JB test was used to determine the choice of resource usage prediction model. As mentioned earlier, ARIMA is used for forecasting when a variable is normally distributed, otherwise, ANN is used. Autocorrelation for CPU and memory for lag value ranging from 5 minutes (lag 1) to 2 hours for VM 01, VM 45 and VM 172 were computed and all were very low except for lags that are multiples of 3 minutes (lag 3, lag 6, lag 9,…), which had a value of 0.3 to 0.4.

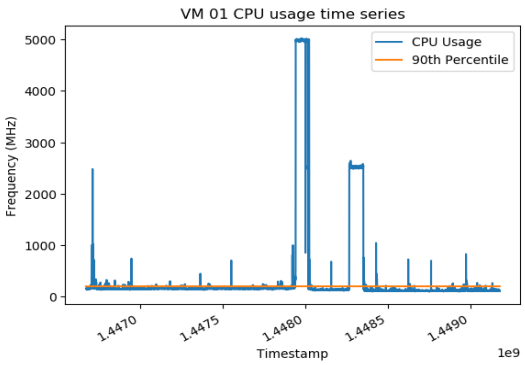


Figure 6.4: A plot of CPU usage showing 90th percentile for VM 01

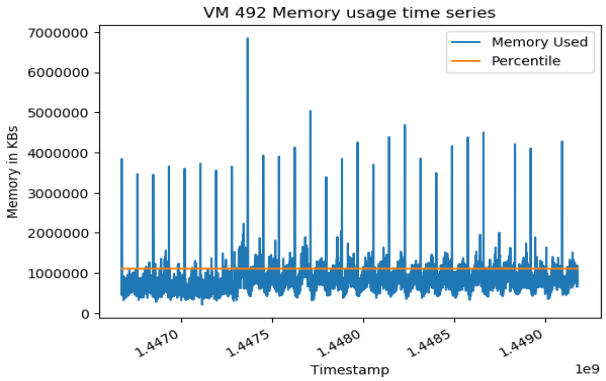


Figure 6.5: A plot of memory usage showing 90th percentile for VM 492

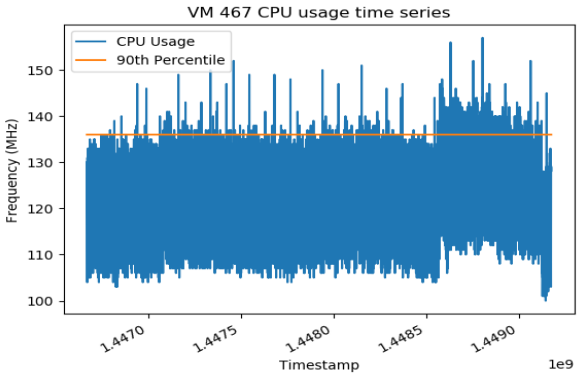


Figure 6.7: A plot of CPU usage showing 90th percentile for VM 467

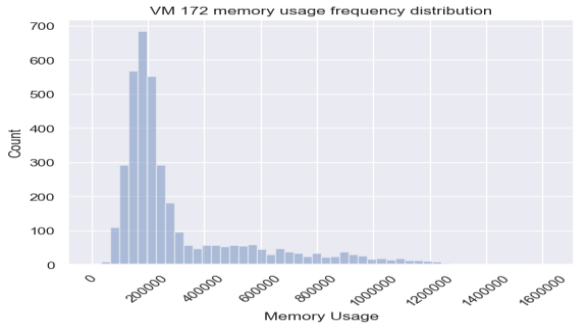


Figure 6.6: Frequency distribution of memory usage for VM 172

Table 6.1: Jarque-Bera (JB) test results for memory usage of VM 172

Item	**Value**
JB	3802.958114
p value	0
Alpha	0.05

6.4 Target cloud model and proposed system architecture

The target cloud model in this section was similar to that used in Chapter 5.2 as shown in Figure 5.1. However, the objective of this section was to resize the VM to match the resources need of the application running in the VM because, more often, cloud users overprovision these resources. Similarly, it was assumed that the CSP has put in place an effective real-time system for monitoring VM resources usage. Based on the results on workload analysis, an architecture for VM sizing in IaaS multi-tenant public cloud based on historical resource usage was thus proposed as shown in Figure 6.8. Because of the dynamic nature of cloud workloads, static resources assigned to VM can never be accurate. The fixed thresholds resource values can trigger unnecessary migrations, which will, in turn,

increase energy consumption (Urul, 2018). Therefore, the proposed architecture predicts resource usage so that, at VM peak resource usage, resources from host reserves can be provisioned. The goal of the architecture was to achieve efficient use of server resources. Energy is reduced due to reduced active servers. Moreover, the customer's monthly bill is reduced because the customers pay for what they use. The architecture components (left side) were separate from the core data center infrastructure (right side) to avoid unnecessary overhead on the core data center resources as discussed in section 6.1 in ARMS characteristics. The components used in the proposed architecture were borrowed from Google's VM right-sizing service and AWS's CloudWatch (Google, 2018; Amazon, Web Services, 2018). For instance, both AWS's CloudWatch and Google's VM right-sizing services collect historical VM resource usage for analysis and when a new VM size proposal is reached at, it is put into a database for access by VM owners. If VM owners need to resize their VMs, they will use VM templates provided by the CSP. The proposed architecture was to make the process of VM sizing automatic (without human intervention) but at the same time give the VM owners (customer) access to right-sizing recommendations. For this reason, billing needs to be adjusted. The proposed VM's components are explained as follows:-

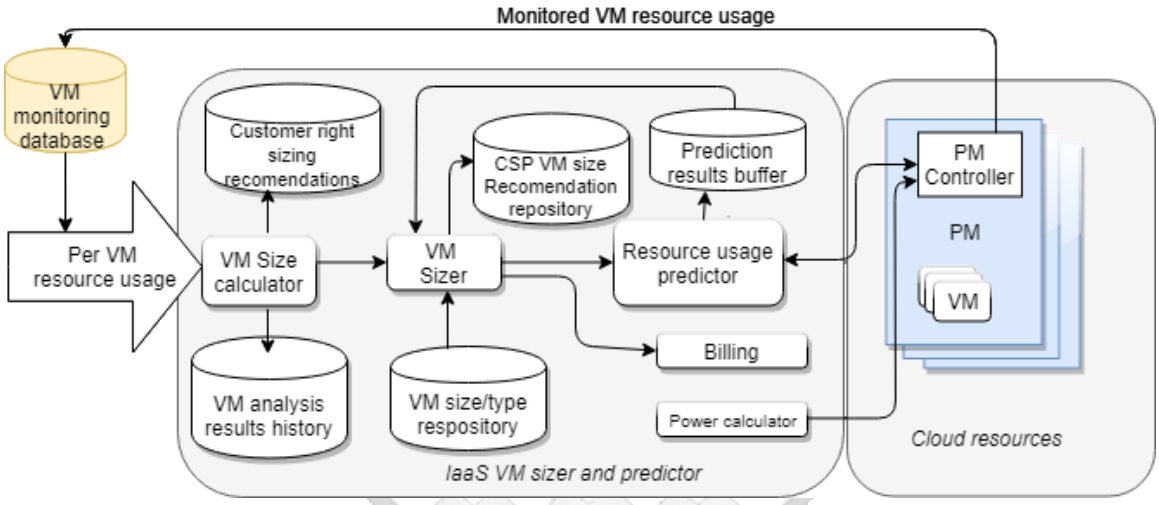


Figure 6.8: Proposed architecture for VM sizing and resource usage prediction

1) *VM size calculator:* this component is the entry point to the *IaaS VM sizer and predictor*. It receives resource usage harvested from a VM via the VMM layer. The resource usage is read and recorded at time intervals thus this data is treated as a time series data. This component analyzes VM resource usage and then determines the right size of the VM (VM fixed size). VMware KB threshold setting (discussed in section 3.3.2) has been adopted for determining the fixed resource usage for VM. CPU usage ceiling $CPU_{ceiling}$ and memory usage ceiling, $MEM_{ceiling}$ are set to represent the percentage of resource usage instances that are below $MEM_{ceiling}$ % and $CPU_{ceiling}$ % for memory and CPU respectively (i.e. percentile). In other words, $MEM_{ceiling}$ is the memory usage whose value is equal to or less than $MEM_{ceiling}$ % of VM memory demands during a period under consideration and is similar to $CPU_{ceiling}$ (i.e. percentile). The resource warnings for these resources are set at 5% above the ceiling. The percentile rankings, R_{cpu} and R_{ram}, for the effective VM size is given in equation 6.3 and 6.4 respectively;

$$R_{cpu} = \left\lceil \frac{CPU_{ceiling}}{100} * N \right\rceil \quad 6.3$$

$$R_{ram} = \left\lceil \frac{MEM_{ceiling}}{100} * N \right\rceil \quad 6.4$$

where N is the total number of observations for either CPU or memory. The ranking, R_{cpu} *and* R_{ram}, are then used to get values, which are treated as 80% of effective fixed VM sizes. This is adapted from the data center Maturity Model (data center Maturity Model uses 60 % but this study uses 80%) discussed in section 3.3.2. We have preferred to consider a ceiling for R_{cpu} *and* R_{ram} because resource warning is set above resource usage ceiling and not below and thus this is an advantage. Thus, if a function *f(.)* gives the values at rank R_{cpu} *and* R_{ram}, then the new effective values for VM CPU, *CPU,* and memory, *MEM*, are given in **equations 6.5** and **6.6** respectively.

$$CPU = \frac{5}{4} f\left(R_{cpu}\right) \quad 6.5$$

$$MEM = \frac{5}{4} f\left(R_{ram}\right) \quad 6.6$$

2) *VM analysis results history*: this is a database that stores the results from VM resource usage analysis.
3) *Customer right-sizing recommendations:* after VM size calculator generates new VM sizes, they are stored in this component for the customer to access. This is a good practice so that customers can understand the changes in size that happen in their VMs.
4) *VM sizer:* this is a very important component in this architecture because it is responsible for implementing VM size recommendations from *VM sizer*. Recommended VM sizes are used to resize a particular VM according to algorithm 6.2 when the recommended size is not equal or close to current VM size. Moreover, this functionality can be delegated to already mature

hypervisors in IaaS space such as Xen, VMware and KVM to make the process automatic. A VM type to be used to resize a current VM is considered close to it if its size is not more or less than 5% of the current VM size. The algorithm then checks in the CSP VM type repository for a VM whose size is equal or close to the recommended size. If such a size is found, it is used to adjust the current VM, otherwise, the current VM is adjusted with the recommended size in which case the *VM sizer* has to update the *CSP VM size recommendation repository*. If memory or vCPU core addition or removal is required, we have proposed the use of CPU hot-plug for CPU and dynamic memory management for memory which has been explained in the work by Tesfatsion (2018). In case a change of full vCPU core is not necessary, the use of per-core DVFS is proposed. The required CPU core frequency DVFS process, f_{core}, is determined according to equation 3.2 (Kamga, 2013). Any time a VM size is adjusted, billing for that VM is also adjusted.

Algorithm 6.2: VM sizer operation

Input: $CalculatedVMSize_{(cpu,ram)}$, $CurrentVMSize_{(cpu,ram)}$
1. **if** $CalculatedVMSize_{(cpu,ram)}$ *approx.* equals $CurrentVMSize_{(cpu,ram)}$ **then**
// *approx.* means if $CalculatedVMSize$ equal to ±5%.
2. **return**
3. **else**
4. auto-scale $resource_{(ram)}$ using memory-ballooning
5 auto-scale $resource_{(cpu)}$ using hot-plug
6. store new recommendations in *CSP VM size recommendation repository*
7 adjust billing
8 pass $CalculatedVMSize_{(cpu,ram)}$, to resource usage predictor
9. **end if**
10. **return**

5) *VM type/size repository*: this is a database of all VM types that have been preconfigured by the CSP. The customer picks a VM type for this repository the first time they request VM creation. This component is used by the *VM sizer* according to **algorithm 4.3.**

6) *CSP VM size recommendation repository:* if the *VM sizer* cannot get a VM type/size from *VM type/size repository* as recommended by *VM size calculator*, it will recommend that this VM type is availed. This recommendation will be stored in this repository for CSP's access. If the CSP decides to create the recommended VM type, it will then be stored in the *VM type/size repository*.

7) *Resource usage predictor*: this is another critical component in the architecture. The fixed VM size recommended by the *VM size calculator* and accomplished by the *VM sizer* can be detrimental to QoS if VM resource demand exceeds the resources ceiling. This is not uncommon owing to the dynamism of the cloud workloads. In the next subsection, an artificial neural networks (ANN) based model is used to show how this component predicts future resource usage so as to provision resources at peak times when VM resources exceed their fixed resources. The aim of this component is to solve a problem, which can be stated as follows; *for a VM, given a time t and a history of VM resource utilization before t, one can forecast resource utilization of the VM at time t.* This study focuses on predicting two resources, CPU and memory because the shortage of these resources during a short period impacts QoS negatively. The CPU capacity, $CPU_{add_{(t)}}$, which needs to be added to the VM at time *t* is determined according to **equation 6.7**.

$$CPU_{add_{(t)}} = CPU_{demand_{(t)}} - CPU_{fixed} \qquad 6.7$$

Where $CPU_{demand_{(t)}}$ is the total CPU capacity required by the VM at time *t* and CPU_{fixed} is the fixed CPU capacity of the VM. This equation applies to memory in a similar manner. Prediction results are stored in a Prediction results buffer. The ANN model is discussed in detail in chapter 6.5.

8) *Billing*: this component is used to calculate and report the cost of running a VM. It follows the pay-as-you-go model. The cost of running a VM, C_{total}, is determined according to **equation 6.8**.

$$C_{total} = C + \left(\alpha_{(ram,cpu)} - \beta_{(ram,cpu)}\right) + \mu \qquad 6.8$$

where C is the cost of the fixed size VM, $a_{(ram,cpu)}$ is the cost of resources borrowed at spiky times, $\beta_{(ram,cpu)}$ is the cost of resources lent to other VMs and μ are other costs associated with the VM such as choice of operating system, data center region, extra services like use of snapshot and multiple Internet Protocol (IP) address.

9) *PM controller*: this component runs in the physical machine and is responsible for monitoring the resource usage of the VMs via the virtualization layer and stores it in the *VM monitoring database*. It is not possible to run it in each VM since the CSP is prohibited from accessing customer VM and that resource performance counter is not virtualized. This component can also give the *resource usage predictor* and *power calculator* access to PM real-time resource usage statistics in the PM. For instance, the *power calculator* may request the current CPU utilization for power usage calculation as shown in equation 5.2.

10) *Power calculator:* this is a simple component that estimates the power consumed by all active hosts at any given time *t* during the execution of the application. Total power is given by a model shown in **equation 5.2**.

11) *Prediction results buffer*: this component is used to hold output results, which are prediction values from the *Resource usage predictor*. A prediction model, such as the one proposed in section 6.5 (equation 6.10) can also be saved in

this component. The contents of this component are accessible by the VM sizer to implementing VM size.

6.5 Artificial neural network (ANN) model for predicting VM resource usage

ANN was a suitable method for prediction in this case because it was shown that the variables (resource usage) had non-Gaussian distribution. ANN has also shown good performance than other prediction models as concluded by various researchers (Ullah, Hassan, & Khan, 2017; Duggan et al., 2017). Moreover, ANN has a great ability to model a non-linear function thus able to handle complex time series (Khashei & Bijari, 2010; Xu et al., 2016). The ANN architecture used in this study had three layers, which were input layer, a hidden layer and an output layer as shown in Figure 6.9.

1) Input layer: In this case, the input was a multivariate time series – each input variable was a time series. These attributes were contained in VM data in the dataset in ARFF format (WEKA's primary data format). The inputs used included CPU usage (x_{cpu}), memory usage (x_{mem}), Disk read throughput (x_r) , disk write throughput (x_w), Network received throughput (x_{neti}) and network transmitted throughput (x_{neto}) represented as $x_{inputs} = \{ x_{cpu}, x_{mem}, x_r , x_w, x_{neti}, x_{neto}\}$.). Feature reduction was accomplished by using WEKA machine learning tool taking *CPU usage* as the target class (Waikato University , 2018). WEKA was the preferred tool because of its easy to use Graphical User Interface (GUI) and stability of its algorithms. The least correlated attributes to the target class were eliminated, which are; CPU capacity provisioned, memory capacity provisioned, disk size, memory usage %, and CPU usage %. A bias node is also included in this layer to ensure that the network will be able to fit the data by avoiding null values. Ordinarily, ANN can determine weak feature via weighting. However, in this study, feature selection was carried out to reduce the time an ANN uses to

determine the weights as well as improve accuracy by weeding out irrelevant features from expert opinion (Iqbal, 2012).

2) Hidden layer: the hidden layer serves to improve prediction accuracy and consists of nodes with activation functions. According to (Karsoliya, 2012), the number of nodes in the hidden layers should be less than twice the number of nodes in the input layer. According to this, an ANN architecture needs to have at most *2p-1* nodes, where *p* is the number of input nodes not including bias. However, in this study, the process of deciding the number of nodes in the hidden layer to the machine software we used was determined by WEKA.

3) *Output layer*: the output layer is used to produce outputs from the ANN model.

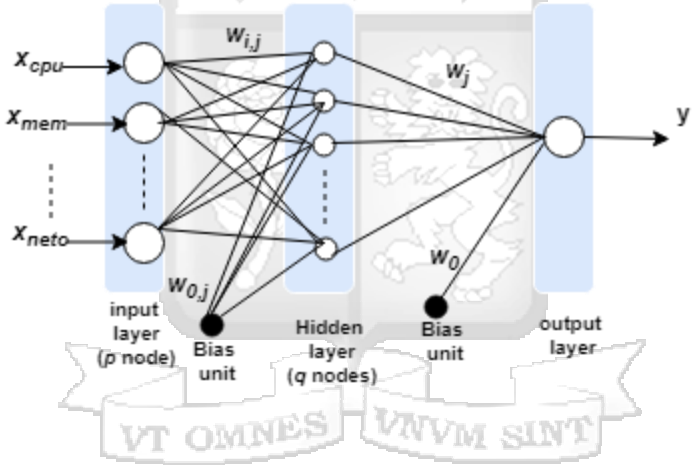


Figure 6.9: Neural network architecture

From the above inputs, the output, O_j of node j in the hidden layer at time t, is given by;

$$O_{j,t} = g\left(w_{0,j} + \sum_{i=1}^{p} w_{ij} \,.\, x_{input_t}\right) = \frac{1}{1 + e^{-\left(w_{0,j} + \sum_{i=1}^{p} w_{ij} . x_{input_t}\right)}} \qquad 6.1$$

Thus, the relationship between the output, y_t, at time t, and input is given by;

$$y_t = w_0 + \sum_{j=1}^{q} w_j \cdot g\left(w_{0,j} + \sum_{i=1}^{p} w_{ij} \cdot x_{input_{t-i}}\right) + \varepsilon_t \qquad 6.2$$

where $g(.)$ is a logistic function, ε_t is the error of the model, $w_{ij}(i=1,2,\ldots p, j=1,2,\ldots q)$ $w_j(j=0,1,2,\ldots q)$ are connection weights, which are parameters of the model, p is the number of input nodes and q is the number of hidden nodes. Specifically, $w_{0,j}$ is the bias included to the input nodes and w_0 is the bias added to the output of hidden nodes. The initial values to the parameters represent a state of knowledge. The input data was partitioned so that 90% of it is used for training and 10% for testing. In the process of partitioning the data, the temporal order of observations was preserved. Thus, VM data rows that belong to either the training set or testing set could not be determined randomly. The predicted values in the ANN model were tested on 4 accuracy metrics i.e. Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE) and Success Rate (SR). SR is the percentage of all predictions, which are equal to or greater than the actual value.

6.6 Evaluation of virtual machine sizing algorithm

This section presents the evaluation procedure of the VM sizing algorithm, together with a discussion of the evaluation results.

6.6.1 Virtual machine sizing algorithm evaluation process

In order to evaluate the performance of the VM sizing approach, CloudSim Plus cloud simulator was used for simulation with GWA-T-13 Materna dataset. The data center configuration shown in Table 6.2 was used to run the evaluation simulations. First, the cloud workloads in the dataset are executed before VM sizing using various VM allocation algorithms – FF, BF, WF and our proposed VM allocation algorithm. Energy consumption was recorded for each algorithm. After VM sizing, the total resource requirements for the data center were calculated,

which were then used to configure the data center shown in the second column of Table 6.2. Similarly, energy consumption was recorded for each of the algorithm used.

Table 6.2: Cloudsim Plus data center configuration before and after VM sizing

Item	**Before VM sizing**	**After VM sizing**
No. of hosts	49	28
No. of VMs	520	520
No. of CPU cores	1298	535
Memory size (in GB)	6780	4142
Hypervisor	VMware ESX	VMware ESX
No. of cores allocated per VM	Varying (1,2,4,6 and 8)	Varying (1, 2 & 3)
Memory size allocated per host (in GB)	Varying (2,4,8 and 16)	Varying (1, 2, 4, & 6)
Host static power	60 % of host peak power	60 % of host peak power

6.6.2 Virtual machine sizing algorithm evaluation results and discussion

The data shown in Table 6.2 shows that, theoretically, CPU cores required to process GWA-T-13 Materna workloads for the 520 VMs reduced to 535 cores from 1298 cores and memory reduced to 4142 GB from 6780 GB. Consequently, the number of VM hosts in the data center potentially reduced from 49 to 28. The reduction of the amount of resources required to process workloads definitely reduced the fixed cost of running VM workloads in the cloud. This cost is the C component from equation 6.8 assuming that μ does not change after VM sizing and $(\alpha-\beta)$ cancels out within the multitenant cloud from the CSP point of view. In addition, the reduction in the number of hosts in the data center reduces the amount of energy consumed. Figure 6.10 shows the amount of energy consumed

when using WF, BF, FF and our proposed VM allocation algorithms before and after VM sizing.

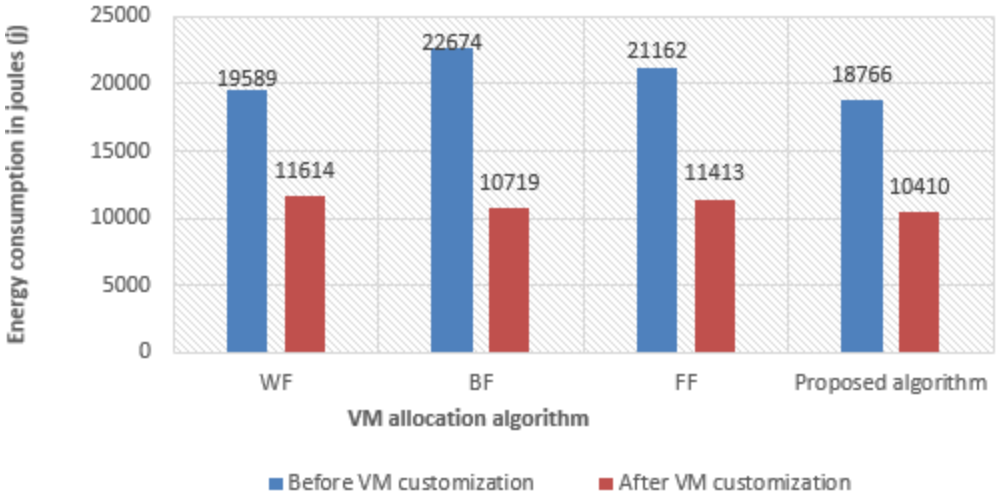


Figure 6.10: A comparison of energy consumption in data center before and after VM sizing using WF, BF, FF and our proposed VM allocation algorithms

It is can be observed that the energy consumed to process similar workload reduced under the different VM allocation algorithms after VM sizing. In fact, a noticeable success is that energy consumed also reduced in the new algorithm that was proposed in Chapter 5. This shows that this algorithm can adapt to different scenarios. These results show that VM sizing has a big potential in managing cloud costs and energy consumption and thus opening up for more opportunities to design more algorithms on the same subject, which can achieve further benefits.

6.7 Evaluation of virtual machine resource usage prediction model

This section presents the evaluation procedure of the VM resource usage prediction, together with a discussion of the evaluation results.

6.7.1 Virtual machine resource usage prediction model evaluation process

To evaluate the ANN VM resource usage prediction model, selected VMs from GWA-T-13 Materna dataset were used on WEKA tool. WEKA is a java based

machine learning tool with a collection of machine learning algorithms that can achieve data preparation, classification, clustering and visualization (Waikato University , 2018). As mentioned earlier, WEKA was preferred as a tool of evaluating resource usage prediction model because of its easy to use GUI and its stable ANN prediction algorithms such as Multilayer Perceptron (MLP). Besides, models created in WEKA can be saved for future reference. In WEKA, one can use either the graphical user interface (GUI) option or the command-line tool.

In this study's evaluation, WEKA's GUI option was used to train a model using our dataset using the following parameters; the number of hidden layers = 1, learning rate = 0.3, momentum = 0.2 and training time or epoch = 1000. As mentioned earlier, all csv data format were converted to WEKA's ARFF format. Feature reduction was achieved on WEKA using the correlations and *CPU* usage as the target class. The least correlated attributes to the target class were eliminated. As a result, CPU capacity provisioned, memory capacity provisioned, disk size, memory usage %, and CPU usage % were eliminated (see Appendix B.2). The ANN model was trained using 90% (7515 instances) of available data and evaluated on 10% (835 instances) of the data. The performance of the model was tested on 4 performance metrics (MAE, MAPE, RMSE and SR) on selected VMs (VM 01, VM 172, VM 405, VM 467 and VM 492).

6.7.2 Virtual machine resource usage prediction model evaluation results and discussion

Table 6.3: ANN prediction performance metrics on VM 01, VM 172, VM 405, VM 467 and VM 492

VM No.	Performance metrics	Resource considered	
		CPU	Memory
01	MAE	23.1	183012.5
	MAPE	13.8	17.9
	RMSE	109.4	219973.7
	SR	61.1	72.5
172	MAE	6.7	142359
	MAPE	5.8	58.9
	RMSE	17.7	201971
	SR	31	39.8
405	MAE	198.9	518969.5
	MAPE	9.5	24.6
	RMSE	265.8	742942.4
	SR	87.3	36.6
467	MAE	7.5	106509
	MAPE	6.1	28.7
	RMSE	13.4	143803
	SR	43	57.6
492	MAE	34.7	158126.8
	MAPE	31.9	20.4
	RMSE	66.1	232459.2
	SR	90.4	67.7

Table 6.3 shows the results of ANN resources prediction model evaluation metrics. For all the VMs that were considered, the MAPE was below 32%, which shows a good prediction performance considering the dynamic nature of the cloud. In fact, the MAPE of the model for CPU on VM 01, VM 172, VM 405 and VM 467 is below 14%, which is impressive. The MAE for the prediction memory showed a good performance – the values for memory seemed bigger but is

attributable to the memory unit of measurement used, which was Kilobytes. This was also shown in RMSE and MAE, which appeared to be bigger in memory. From the values of performance metrics shown in Table 6.3, it was concluded that ANN model achieved a good level of success on the highly dynamic cloud workloads.

A higher performance metric was not a problem in this study because it was a measure of the degree to which a predicted value is bigger or smaller than the actual value. This model was more concerned when a resource demand predicted was lower or equal to the actual value because QoS was then affected. Thus, SR was computed, which was an important metric because it measured the degree to which a VM was likely not to suffer from low QoS because of lack of resources. For VM 01, VM 175, VM 405, VM 467 and VM 492, the chance that the VMs were unlikely to suffer from poor QoS was 61.1%, 31%, 87.3%, 43% and 90.4% respectively. Even if a VM was likely to suffer from low resource provisioning, such as those with low SR, statistical multiplexing came in handy, where, if a VM needs more resources than was predicted, it could use resources from co-resident idle VMs. The analysis presented in algorithm 6.1 demonstrates that statistical multiplexing was a potential technique.

Graph plots that visually compares predicted and actual values of CPU and memory for different VMs are as shown in Figure 6.11 to Figure 6.16. The graph plots only showed a portion of the predicted values with the order of observations preserved. The results in the figures and those from Table 6.3 showed that the model performed differently on the different VMs. This means that each VM's model needs to be generated separately.

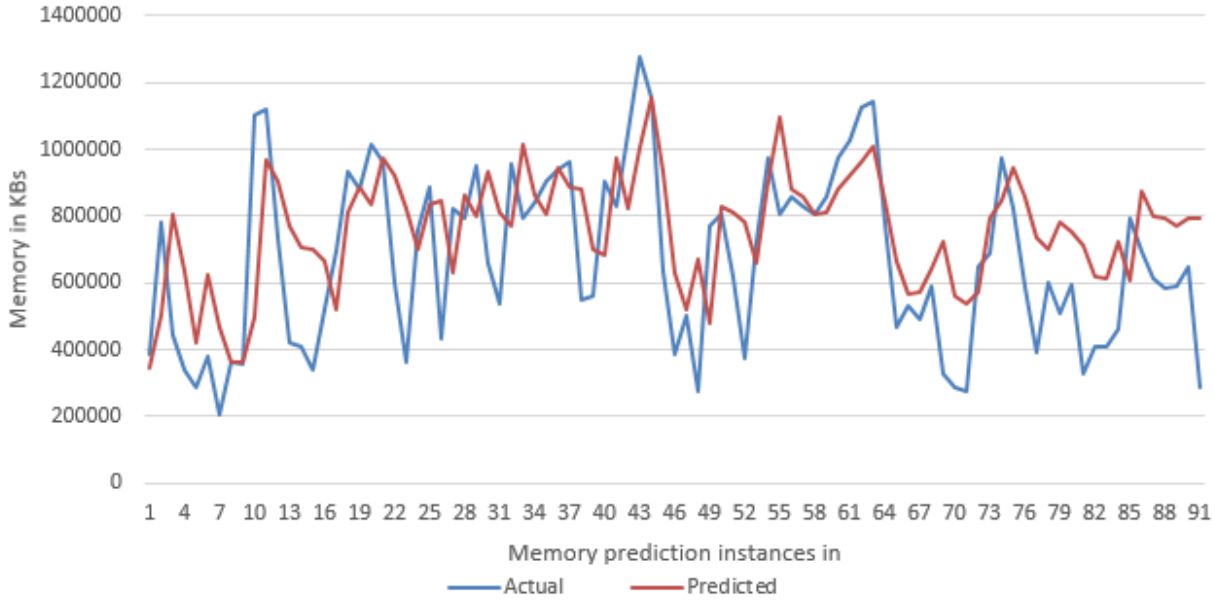


Figure 6.11: Graphical representation of a comparison between predicted and actual values for memory for VM 01

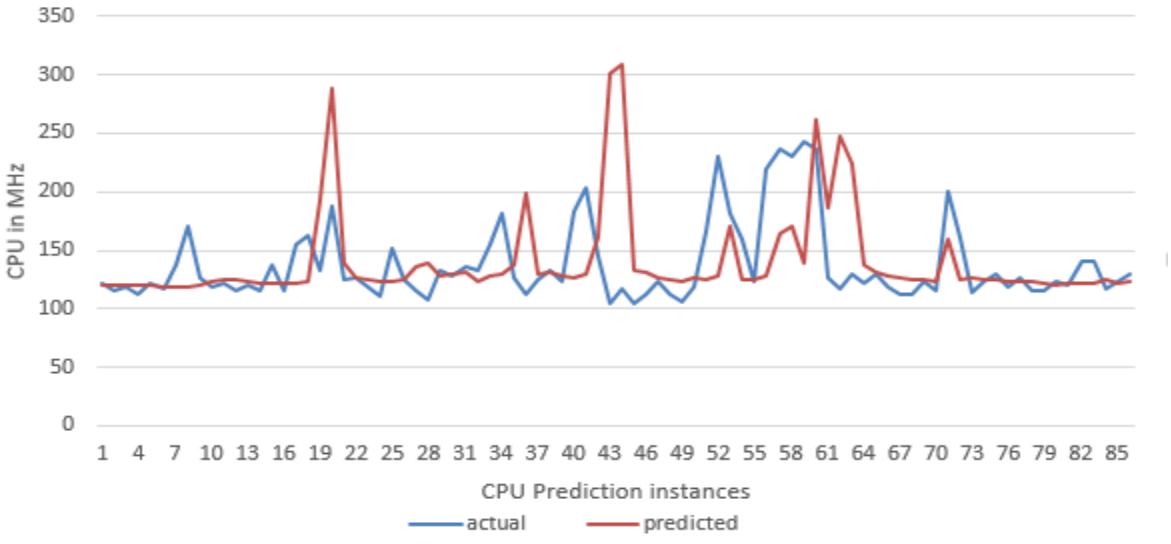


Figure 6.12: Graphical representation of a comparison between predicted and actual values for CPU for VM 01

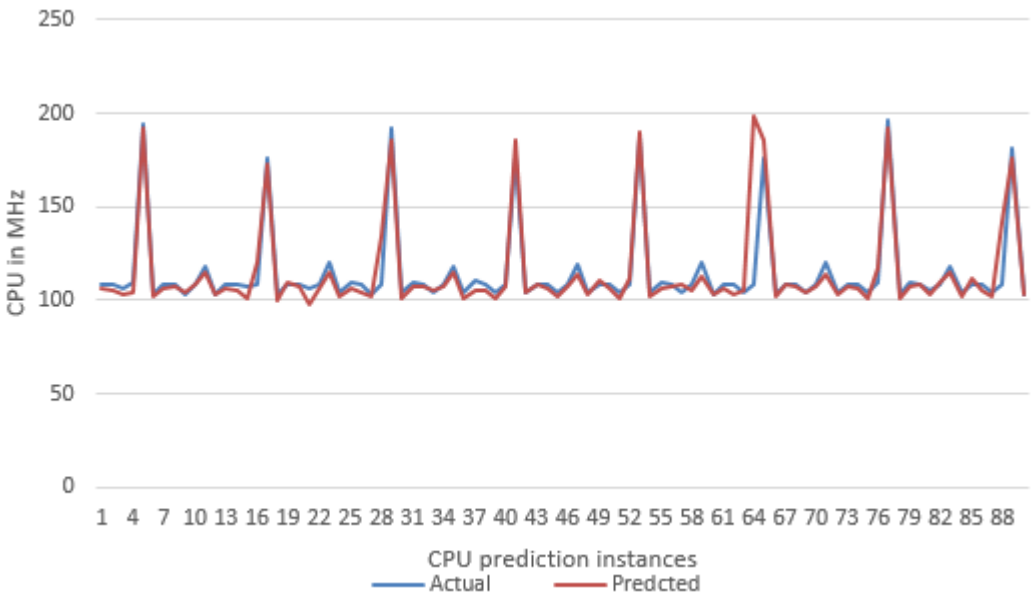


Figure 6.13: Graphical representation of a comparison between predicted and actual values for CPU for VM 172

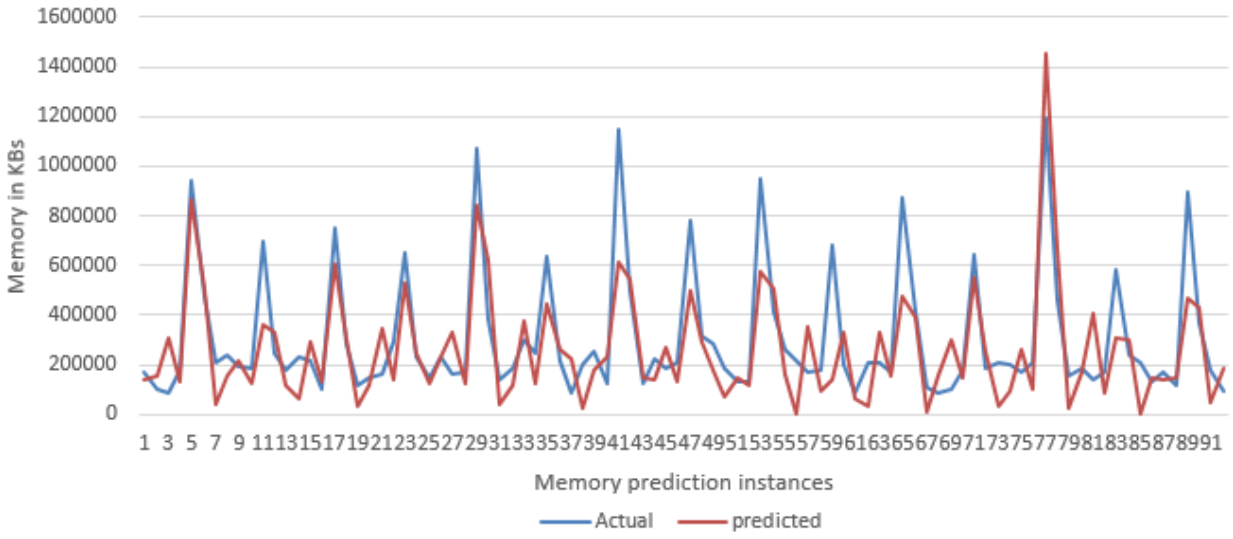


Figure 6.14: Graphical representation of a comparison between predicted and actual values for memory for VM 172

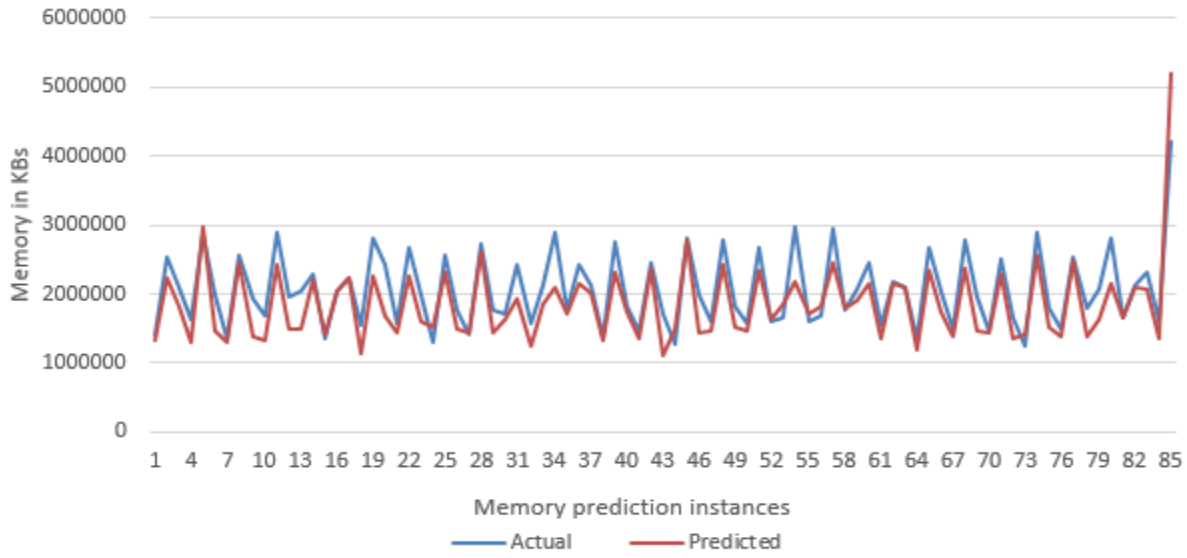


Figure 6.15: Graphical representation of a comparison between predicted and actual values for memory for VM 405

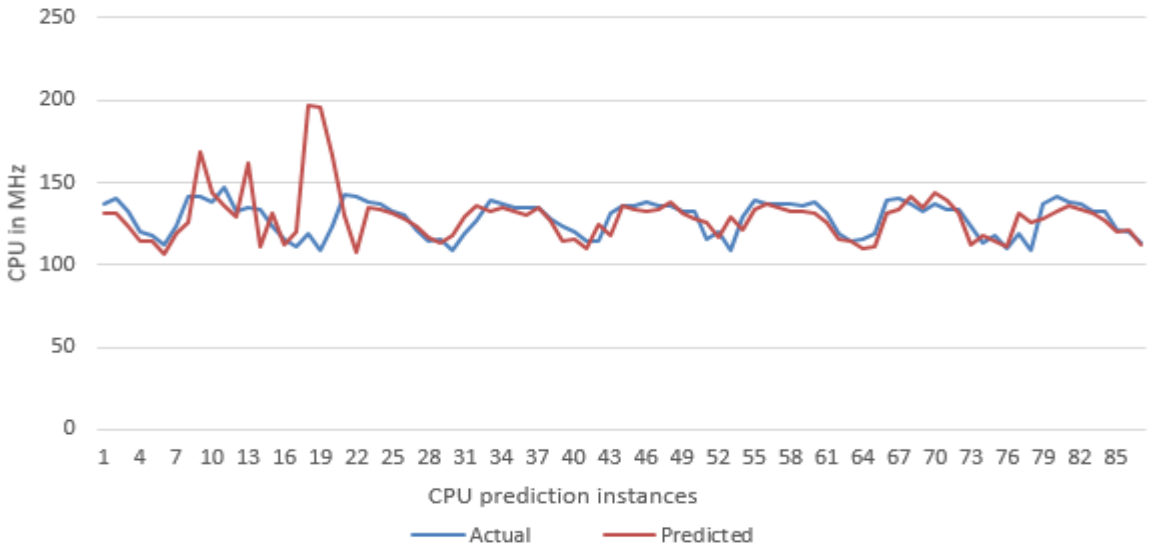


Figure 6.16: Graphical representation of a comparison between predicted and actual values for CP for VM 405

6.8 Conclusion

In this chapter, an architecture for VM sizing and VM resource usage prediction was proposed. The main goal was to determine a fixed size for a user VM, which matches the user's application resource demands. Because of the dynamism of cloud workloads, allocating a VM a fixed size without overprovisioning is dangerous because it will lead to deterioration of QoS service when VM resource demands exceed the provisioned resources. This happens during peak times. For this reason, an ANN model for predicting resource usage for a VM was proposed, such that more resources can be availed during peak times. From the analysis of real cloud workloads, it was shown that indeed, resources allocated to VMs go to waste and thus VM resizing technique is of value. It was also shown that cloud workloads resource demands vary greatly and thus fixed VM sizes is not a complete solution. Further, from the workload analysis, was shown that a big opportunity to exploit statistical multiplexing exists since many VMs do not peak simultaneously.

The evaluations conducted on the proposed VM sizing architecture indicated greater success. For instance, the VM sizing architecture successfully reduced the amount of resources required to process application workload, which in turn reduced the number of required hosts thus reducing energy consumption (Table 6.2 and Figure 6.10). Further, the metrics used to evaluate the VM resource usage prediction using an ANN model showed that greater accuracy can be achieved while forecasting future VM resource usage for provision.

Chapter 7

Conclusion and Future Research Directions

This chapter summarizes the study's investigation on Energy-Efficient Resources Utilization Algorithms for Cloud data center Servers and its main outcomes. It also identifies open research challenges, which this study did not address and future research directions. The first part summarizes the main contributions and conclusions and the second part presents future research directions.

7.1 Summary

Currently, cloud computing is a new norm because it is cheap, a pay-as-use service. As organizations continue to adopt it across the world, cloud providers are putting up more data centers to ensure Quality of Service (QoS). Unfortunately, data center servers consume a lot of energy. As a result, cloud providers experience high operating costs (electricity bills), which reduces profits, increases Total Cost of Ownership (TCO) of data center infrastructure and increases carbon dioxide emission.

This research has shown that, among the many causes of energy wastage in data center servers, low server utilization tops the list. Low server utilization means that a data center needs more servers to process workloads and that idle power of servers is wasted. This research then embarked on addressing this problem. In this respect, experiments were conducted to determine consolidation friendly workload combination by investigating the power and performance profiles of the various computing resources (CPU, memory and hard drive) on heterogeneous and homogeneous workloads. Based on the profiling results, an energy-aware VM allocation policy was designed, developed and evaluated. Further, to address the problem of resource wastage in multi-tenant IaaS cloud,

which results in energy wastage, a VM sizing algorithm was designed and evaluated. The VM sizing algorithms were extended to include VM resource usage prediction to take care of the bursty nature of cloud workloads. In each of chapter 2-6, a number of conclusions have been made.

Chapter 2 reviewed cloud computing, how energy is consumed in data centers and data center servers and identified the factors. On the other hand, Chapter 3 reviewed the existing techniques that are currently used for efficient energy utilization. The following conclusions were made;

a) IaaS public cloud has the highest potential of being adopted because it is the choice of big and small organizations, who want to take full control of their VMs. It validates Bezos' law. Yet this service model poses the highest threat to cloud resource wastage, which in turn leads to energy wastage.
b) The causes of energy wastage in data center are many but those that present the highest opportunity to reduce this wastage when they are addressed are low server utilization and workload interference as a result of resource sharing by co-residency VMs. Low server utilization causes CSP to use more servers to process workloads and wastage of idle power of servers. Workload interference causes workload processing to take longer hence consuming more energy – refer to equation 2.2
c) There are two classes of techniques that can be used to address the problem of energy efficiency – software techniques and hardware techniques. Software techniques relate to the behavior of application workloads and hardware techniques relates to frequency and voltage scaling on computing resources.
d) Hardware techniques of efficient energy utilization, such as DFVS, are getting overwhelmed due to the recent development of processor

technologies that have saturated clock frequencies and developed better sleep modes. Other hardware components such as memory have narrow dynamic power ranges than static power. Thus, application-based energy efficiency techniques present the biggest opportunity of addressing energy wastage in data center servers.

e) Cloud workload characterization is a very important way of understanding cloud workloads and is mostly dominated by statistical techniques. In fact, more time and effort should be spent on understanding the behavior of applications than creating new energy efficient algorithms.

f) The analysis of publicly available real backend workload trace logs such as GCT was done exhaustively and it is time to analyze other newly available cloud workloads such as Delf University's grid workload archive and Alibaba cluster trace (Cheng, Chai, & Anwar, 2018). New cloud workloads can present new findings because they have been executed on new server hardware and more recent software such as operating systems and hypervisors.

Chapter 4, 5 and 6 presented the design and evaluation of the proposed algorithms to address the problem described in Chapter 1.2. Power and performance profiles of the various computing resources (CPU, memory and hard drive) on heterogeneous and homogeneous workloads were investigated through experiments conducted. A VM allocation algorithm was developed, which ensures that VMs with similar resource usage profiles are not co-located. This algorithm was implemented and evaluated on CloudSim Plus cloud simulator. The proposed algorithm outperforms FF, BF and WF VM allocation algorithms. A VM sizing algorithm was also proposed, which ensures that VM resource overprovisioning by IaaS cloud users is addressed. The VM sizing algorithm determines a fixed VM size, which is complemented by VM resource usage prediction to provision

resources when VM demand exceed fixed VM sizes. The VM resource prediction approach is based on multivariate prediction using an ANN model. VM sizing and VM resource prediction approaches were also evaluated on CloudSim Plus and WEKA respectively. The approaches proposed encompasses end-to-end architectures, which can be used to implement them in modern hypervisors. From these three chapters, the following conclusions have been made;-

a) Workload consolidation limiting factor for CPU intensive workloads and disk intensive workloads is power consumption and performance respectively.
b) VM allocation policies should co-locate heterogeneous workloads and not homogeneous workloads. Homogeneous workloads put a hotspot of activity on particular computing resources and leave other computing resources idle. The intense activity on these computing resources increases interference among the workload due to the competition making workload processing to take more time. Homogeneous workloads thus lead to energy wastage in two ways: 1) idle components use energy without processing any useful workload, 2) long processing time means more energy usage. Therefore, heterogeneous workloads are consolidation friendly.
c) Clustering approaches can achieve great success in characterizing cloud workloads in identifying similar and dissimilar workloads for purposes of designing VM allocation policies. From this standpoint, any VM allocation policy should strive to understand the behavior of cloud workload before allocation.
d) Indeed, VM sizing approach can be used to reduce resource wastage and by extension energy wastage. In this study, a simple approach was used (percentiles) to propose new fixed VM sizes in which case we succeeded in reducing CPU and memory allocation from 1298 cores to 535 core and 6780 GB to 4142 GB respectively.

e) The use of ANN model to predict VM resource usage in the cloud can achieve great success. This is despite the non-linear nature of time series data associated with the resource demands of cloud workloads.

f) Cloud simulation software has matured and can support complex experiments and can be used to draw useful conclusions, which can be applied to real cloud infrastructure. This study successfully tested the proposed algorithms using CloudSim Plus cloud simulator easily, quickly and free of charge.

7.2 Study objectives

7.2.1 How server related factors influence energy consumption and wastage in a cloud data center

A review carried out in this study revealed that there are a number of factors, which influence energy consumption and wastage in cloud data center servers. These factors included the levels of server utilization, idle power wastage, adoption of energy efficient, choice of server utilization metric and data center thermal management. Of all these factors, level of server utilization plays the biggest role in either saving energy or wasting energy. The review showed that servers are generally underutilized despite consuming up to 70% of their peak power. This study, through experimentation, showed that level of server utilization is affected by workload types (homogeneous or heterogeneous) and VM sizes. As a result a VM allocation algorithm and a VM sizing algorithm were proposed.

7.2.2 Techniques used to reduce energy consumption and wastage by cloud data center servers

This study reviewed various techniques, which are currently used to address the problem of energy efficiency in cloud data centers. These techniques included DVFS, server power switching, hardware inbuilt capabilities such SSD

and VM consolidation. However, each of these techniques shows a weakness. For instance, VM consolidation does consider workload types and VM sizes, which are factors that affect level of server utilization. On the other hand, DVFS is designed for processor bound tasks because dynamic power ranges for other computing resources such as memory are narrower. This discovery led to the design of a VM allocation algorithm and a VM sizing algorithm.

7.2.3 Design and development of algorithms, which considers contextual server factors, to reduce energy consumption and wastage in cloud data center servers

In this study, a VM allocation algorithm and a VM sizing algorithm were proposed to address the problem of low level of server utilization or *resource overprovisioning*. The VM sizing algorithm was extended to include VM resource usage prediction to address the problem of *performance unpredictability.* VM allocation algorithm co-locates heterogeneous VMs in the same PM as opposed to homogeneous VMs. VM sizing algorithm resizes user VMs to match the application's resources demands. Both algorithms were implemented on a cloud simulator, CloudSim Plus.

7.2.4 Evaluation of the performance of the proposed algorithms

The algorithms, VM allocation algorithm and a VM sizing algorithm, which were proposed in this study, were implemented on a cloud simulator, CloudSim Plus. The VM allocation algorithm was evaluated by comparing it with well-known FF, BF and WF VM allocations algorithms, which do not consider workload characteristics. Results showed that the proposed VM allocation algorithms outperformed them all. On the other hand, VM sizing algorithm was evaluated on CloudSim Plus simulators by processing GWA-T-13 Materna before and after VM sizing. Results showed that, after VM sizing, energy consumption by data center was reduced.

7.3 Future work

The contributions that were achieved made in this study are useful. However, some issues related to energy efficiency as a result of efficient resource utilization in cloud data centers have not been addressed. These limitations will be addressed in future work.

7.3.1 Server power model

In this study, a power model that only considers the processor as the one that controls the use of power was adopted. The model is as sown in equation 2.3. The main assumption made is that the processor, among other computing resources, shows the highest range of dynamic power. However, from the experiments conducted on power and performance profiles of various computing resources, it was concluded that other computing resources consumes power and need to be considered in the development of a comprehensive server power model. Although the dynamic power ranges of the other resources are narrower, their static power can contribute to power consumption. As future work, we wish to develop and apply a power model, which considers all computing resources as a potential consumer of power supplied into a server.

7.3.2 Design of VM Allocation Policy

In this study, an architecture for workload harvesting, workload characterization, which includes a VM allocation algorithm was proposed. The evaluations carried out have shown that the algorithms can achieve energy savings at least on the cloud workloads that were used. The clustering approach for the purposes of co-locating dissimilar workloads has considered only the processor and memory. The assumption is that these two resources are very sensitive to QoS when they are unavailable for a short time. However, from the experiments of determining performance profile on computing resources, it was discovered that the other resources such as the hard drive are very sensitive to

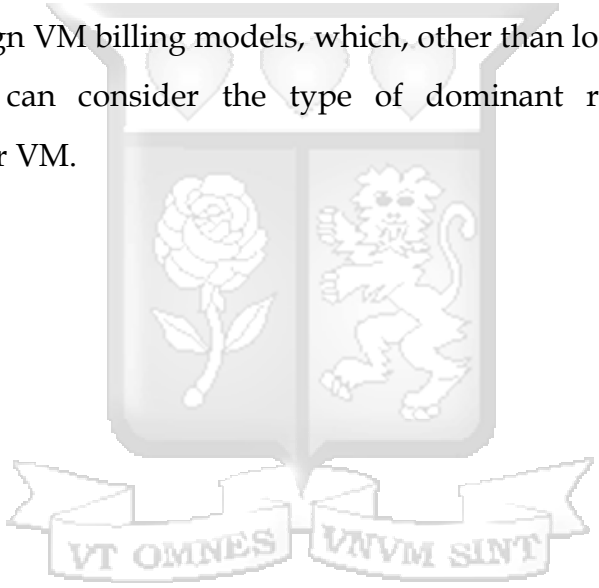
resource contention caused, which is caused by sharing of resources. In fact, the slowness of disk I/O can cause the processor to remain idle, which causes low resource utilization. Thus, in the future, the design of any VM allocation policy needs to consider the interference caused as a result of all the computing resources before VM allocation. Moreover, there are other factors not related to VM resources, which may affect the performance characteristics of VMs. For instance application task frequency, task density or submission rate and task length are examples of such factors. Unfortunately, it is difficult to land on publicly available cloud workloads with all these properties. For instance, GWA-T-13 Materna, which was used in this study have VM resource usage information but lack application task information. On the other hand, GCT has rich information on application task properties, but lack VM information. In future work we will focus on testing VM allocation solutions on a variety of cloud workloads or solicit for cloud workloads, which have all the properties that have been mentioned. Organizations, which manage private clouds are thus challenged to publish cloud workloads that have rich properties because they have control of their infrastructure from PM to VM and virtualization layer. Testing of VM allocation algorithms on a variety of workloads can also help to test how the designed algorithms scale to other infrastructure.

7.3.3 Implementation of the proposed algorithm

Another important future work in this study is to implement the proposed algorithms as part of real cloud software such as OpenStack. The implementations in study was done a cloud simulator thus were unable to test the proposed algorithm on real cloud infrastructure. Implementing these algorithms as part of OpenStack will enable the execution of the algorithms on real cloud infrastructure.

7.3.4 Advanced VM Sizing and Advanced VM Billing

Virtualization helps in the execution of VMs in one server. In IaaS, it has been observed that VM sizing can affect energy consumption and costs of running VMs in the data center. Therefore a solution has been designed, which has achieved a good level of success. We still believe that further investigation is needed to mint more benefits from VM sizing in terms of energy savings and reducing VM hosting costs. For instance, the history of resource borrowing among multi-tenant VMs can be used as an input to VM sizing. It was also observed that VM sizing can save on the costs of hosting VMs in data centers. As future work, we hope to design VM billing models, which, other than looking at a fixed cost of VM resources, can consider the type of dominant resources and energy consumption per VM.



References

Abbas, O. M. (2015). Neural Networks in Business Forecasting. *International Journal of Computer, 19*(1), 114-128.

Akhil, G., & Navdeep, C. (2015). A Proposed Approach for Efficient Energy Utilization in Cloud Data Center. *International Journal of Computer Applications (0975 – 8887), 115*(11).

Albert, G., James, H., David, A. M., & Parveen, P. (2009). The cost of a cloud: research problems in data center networks. *The ACM Digital Library is published by the Association for Computing Machinery, 39*(1).

Al-Dulaimy, A., Zantout, R., Itani, W., & Zekri, A. (2016). Job Submission in the Cloud: Energy Aware Approaches. *Proceedings of the World Congress on Engineering and Computer Science* . San Francisco, USA.

AlJahdali, H., Albatli, A., Garraghan, P., Townend, P., Lau, L., & Xu, J. (2014). Multi-Tenancy in Cloud Computing. *8th IEEE International Symposium on Service-Oriented System Engineering.* Oxford, UK.

AllAfrica. (2018). *Kenya: Angani Turns Talk Into Business As the Transition to the Cloud Begins to Take Wing*. Retrieved August 1, 2019, from AllAfrica: https://allafrica.com/stories/201812170288.html

Almqvist, O. (2019). *A comparative study between algorithms for time series forecasting on customer prediction: An investigation into the performance of ARIMA, RNN, LSTM, TCN and HMM.* University of Skövde.

Amazon. (2018). *Amazon EC2 Instance Types*. Retrieved December 04, 2018, from Amazon.com: https://aws.amazon.com/ec2/instance-types/

Amazon Web Services. (2018). *Right Sizing: Provisioning Instances to Match Workloads: AWS Whitepaper*. Amazon Web Services, Inc. Retrieved from https://docs.aws.amazon.com/aws-technical-content/latest/cost-optimization-right-sizing/cost-optimization-right-sizing.pdf

AMD. (2017). *AMD A-Series Desktop APUs*. Retrieved January 26, 2017, from AMD: http://www.amd.com/en-us/products/processors/desktop/a-series-apu

Amiri, M., & Mohammad-Khanli, L. (2017). Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications, 82*(1), 93-113.

Anderson, M. (2016). *Intel says chips to become slower but more energy efficient*. Retrieved January 26, 2017, from The Stack: https://thestack.com/iot/2016/02/05/intel-william-holt-moores-law-slower-energy-efficient-chips/

Angani. (2019). *Angani: Processing Africa's Data*. Retrieved August 1, 2019, from Angani: http://angani.co

Anne-C´ecile, O., Assuncao, M. D., Lef`evre, L., & Tut´oi, R. (2014). A Survey on Techniques for Improving the Energy efficiency of large scale distributed systems. *ACM Computing Surveys (CSUR), 46*(4).

Anton, B. (2013). *Energy-Efficient Management of Virtual Machines Data Centers for Cloud Computing*. THE UNIVERSITY OF MELBOURNE, Department of Computing and Information Systems. The University of Melbourne.

Anton, B., & Rajkumar, B. (2010). Energy Efficient Resource Management in Virtualized Cloud Data Centers. *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, (pp. 826-831).

Anton, B., Jemal, A., & Rajkumar, B. (2011). Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Journal of Future Generation Computer Systems, 28*(5).

Ashwin, K. S., Rahul, R., Dheepan, & Sendhil, K. (2015). An Optimal Ant Colony Algorithm for Efficient VM Placement. *Indian Journal of Science and Technology, 8*(S2), 156–159.

Bahga, A., & Madisetti, V. K. (2011). Synthetic Workload Generation for Cloud Computing Applications. *Journal of Software Engineering and Applications, 4*(7).

Bangari, K., & Rao, C. (2016). Real Workload Characterization and Synthetic Workload Generation . *International Journal of Research in Engineering and Technology , 5*(5).

Bankole, A. A. (2013). *Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment.* Ontario, Canada: Carleton University.

Barroso, L. A., & Hölzle, U. (2009). *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale.* Google Inc.

Basmadjian, R., Ali, N., Niedermeier, F., Meer, H. d., & Giuliani, G. (2011). A Methodology to Predict the Power Consumption of Servers in Data Centres.

Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking. New York, USA.

Bronson, A. X., P., R., & Raja, S. S. (2018). A Dynamic Memory Allocation Strategy for Virtual Machines in Cloud Platform. *nternational Journal of Pure and Applied Mathematics, 119*(15), 1423-1444.

Cano, I., Aiyar, S., & Krishnamurthy, A. (2016). Characterizing Private Clouds: A Large-Scale Empirical Analysis of Enterprise Clusters . *SoCC '16 Proceedings of the Seventh ACM Symposium on Cloud Computing .*

Cao, R., Yu, Z., Marbach, T., Li, J., Wang, G., & Liu, X. (2018). Load Prediction for Data Centers Based on Database Service. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC).* Tokyo, Japan.

Carmody, B. (2018). *Infrastructure On Demand Is Giving Small Businesses An Edge*. (Inc) Retrieved OCtober 01, 2018, from https://www.inc.com/bill-carmody/infrastructure-on-demand-is-giving-small-businesses-an-edge.html

Chaima, G. (2014). *Energy efficient resource allocation in cloud computing Enviroment.* Institut National des T´el´ecommunications. Paris, France : Institut National des T´el´ecommunications.

Chang, Y., Gu, C., & Luo, F. (2017). Energy Efficient Virtual Machine Consolidation in Cloud Datacenters . *The 2017 4th International Conference on Systems and Informatics (ICSAI 2017)* (pp. 401- 406). Hangzhou, China: IEEE.

Chen, M., Zhang, H., Su, Y.-Y., Wang, X., Jiang, G., & Yoshihira, K. (2011). Effective VM sizing in virtualized data centers. *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops.* Dublin, Ireland .

Chen, X. (2014). *Failure Analysis and Prediction in Compute Clouds.* University of Science and Technology of China .

Chen, X., Rupprecht, L., Osman, R., Pietzuch, P., Franciosi, F., & Knottenbelt, W. (2015). CloudScope: Diagnosing and Managing Performance Interference in Multi-tenant Clouds. *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems.*

Cheng, Y., Chai, Z., & Anwar, A. (2018). *Characterizing Co-located Datacenter Workloads: An Alibaba Case Study.* George Mason University.

Chilisa, B., & Kawulich, B. (2012). Selecting a research approach: paradigm, methodology and methods. In C. Wagner, B. Kawulich, & M. Garner, *Doing Social Research: A Global Context.* London, United States: McGraw-Hill Education .

Chirag, J. R. (2014). A Survey on Different Virtual Machine Placement Algorithms. *International Journal of Advance Research in Computer Science and Management Studies, 2*(2).

Choudharya, A., Rana, S., & Matahai, J. (2016). A Critical Analysis of Energy Efficient Virtual Machine Placement Techniques and its Optimization in a Cloud Computing Environment. *International Conference on Information Security & Privacy (ICISP2015)* (pp. 133-138). India: Elsevier.

Cisco. (2013). *Cisco.com*. Retrieved from Power Management in the Cisco Unified Computing System: An Integrated Approach: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/unified-computing/white_paper_c11-627731.html

Columbus, L. (2018). *83% Of Enterprise Workloads Will Be In The Cloud By 2020*. Retrieved November 2, 2018, from https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/#5e2f5b256261

Costa, G. D., Grange, L., & Courchelle, I. D. (2016). Modeling and Generating large-scale Google-like Workload. *The Seventh International Green and Sustainable Computing Conference* . Hangzhou, China .

Creswell, J. (2009). *Research Design: Qualitative, Quantitative and Mixed Methods Approaches* (Third ed.). SAGE Publications. Inc.

Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *IEEE Network, 29*(2).

Data-Economy. (2019). *Sustainable Digitalisation, Powered By Sweden*. Retrieved August 1, 2019, from Data-Economy: https://data-economy.com/how-do-we-do-more-with-less/

David, S. S., & Anbuselvi, R. (2016). Autonomic Resource Provisioning Algorithm for Cloud Computing using Match Making Technique. *International Journal of Advanced Research in Computer Science and Software Engineering, 6*(9), 168 - 173.

Deborah, M., Rodrigo, N. C., Rajkumar, B., & Danielo, G. G. (2015). Workload modeling for resource usage analysis and simulation in cloud computing. *Computers and Electrical Engineering, 47*(2015), 69-81. Retrieved from http://www.cloudbus.org/papers/WorkloadMod-Sim-CEE.pdf

Delf University. (2018). *The Grid Workloads Datasets*. (Delf University) Retrieved 2 October, 2018, from http://gwa.ewi.tudelft.nl/datasets/

Delft University of Technology. (2015). *GWA-T-12 Bitbrains*. Retrieved March 5, 2017, from Grid Workload Archive: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

Delimitrou, C. (2015). *Improving Resource Efficiency In Cloud Computing.* Stanford University.

Dhiman, G. (2011). *Dynamic Workload Characterization for Energy Efficient Computing.* University of California.

Dhiman, G., Mihic, K., & Rosing, T. (2010). A System for Online Power Prediction in Virtualized Environments Using Gaussian Mixture Models . *in Proceedings of the 47th Annual ACM/IEEE Design Automation Conference (DAC)*, (pp. 807-812).

Duggan, M., Mason, K., Duggan, J., Howley, E., & Barrett, E. (2017). Predicting Host CPU Utilization in Cloud Computing using Recurrent Neural Networks. *The 8th International Workshop on Cloud Applications and Security.*

Elijorde, F., & Lee, J. (2015). Attaining Reliability and Energy Efficiency in Cloud Data Centers Through Workload Profiling and SLA-Aware VM Assignment. *International Journal Advance Soft Computer Applications , 7*(1).

Esha, B., Y. J., & Biju, I. (2016). Energy Efficient Virtual Machine Placement using Enhanced Firefly Algorithm. *Journal of Multiagent and Grid Systems , 12*(3), 167-198.

Fan, X., Weber, W.-D., & Barroso, L. A. (2007). Power Provisioning for a Warehouse-sized Computer. *In Proceedings of the ACM International Symposium on Computer Architecture,, June 2007*, (pp. 13-23). San Diego, CA, USA.

Fei, X., Fangming, L., Linghui, L., Hai, J., Li, B., & Baochun, L. (2014). iAware: Making Live Migration of Virtual Machines Interference-Aware in the Cloud. *IEEE Transactions on Computers*, 3012 - 3025.

Frey, S., Disch, S., Reich, C., Knahl, M., & Clarke, N. (2015). Cloud Storage Prediction with Neural Networks. *The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization.*

Ganesan, R., Sarkar, S., & Narayan, A. (2012). Analysis of SaaS Business Platform Workloads for Sizing and Collocation. *2012 IEEE 5th International Conference on Cloud Computing (CLOUD.* IEEE.

Gicheru, E. (2013). The psychology of unmarried men in Nairobi: A case study of three bachelors over forty. *African Journal of History and Culture, 5*(6), 126-137.

Gohil, B., Shah, S., Golechha, Y., & Patel, D. (2016). A Comparative Analysis of Virtual Machine Placement Techniques in the Cloud Environment. *International Journal of Computer Applications (0975 – 8887), 156*(14).

Gokul, A., & Priya, S. (2016). Energy Optimization in Cloud Computing by EGC Algorithm. *International Conference on Innovations in Engineering and Technology*, (pp. 10-12). Madurai, India.

Google. (2018). *Applying Sizing Recommendations for VM Instances*. (Google) Retrieved November 1, 2018, from https://cloud.google.com/compute/docs/instances/apply-sizing-recommendations-for-instances

Google. (2018). *GOOGLE APP ENGINE*. Retrieved from Google cloud: https://cloud.google.com/appengine/

Google. (2019). *Moving toward 24x7 Carbon-Free Energy at Google Data Centers: Progress and Insights.* Retrieved from Google APIs: https://storage.googleapis.com/gweb-sustainability.appspot.com/pdf/24x7-carbon-free-energy-data-centers.pdf?utm_source=newsletter&utm_medium=email&utm_campaign=newsletter_axiosgenerate&stream=top

Gorelik, E. (2013). *Cloud Computing Models.* Massachusetts Institute of Technology.

Green Grid. (2010). *The Green Grid Data Center Compute Efficiency Metric: DCcE.* White paper. Retrieved from http://www.thegreengrid.org/~/media/WhitePapers/DCcE_White_Paper_Final.pdf?lang=en

Hadi, G., & Massoud, P. (2016). Achieving Energy Efficiency in Datacenters by Virtual Machine Sizing, Replication, and Placement. In *Energy Efficiency in Data Centers and Clouds.* Elsevier Science.

Han, G., Que, W., Jia, G., & Shu, L. (2016). An Efficient Virtual Machine Consolidation Scheme for Multimedia Cloud Computing. *MDPI:sensors, 16*(2).

Hassan, A. A.-E. (2015). *Workload Characterization, Controller Design and Performance Evaluation for Cloud Capacity Autoscaling.* Ume°a University, Department of Computing Science. Ume, Sweden: Ume°a University.

Hintemann, R., & Clausen, J. (2016). Green Cloud? The current and future development of energy consumption by data centers, networks and end-user devices. *ICT4S Conference.* Amsterdam. Retrieved from https://www.borderstep.de/wp-content/uploads/2016/09/ICT4S-Hintemann-Clausen-Green-Cloud-final-2016.pdf

Hu, R., Liu, G., Jiang, J., & Wang, L. (2015). A New Resources Provisioning Method Based on QoS Differentiation and VM Resizing in IaaS. *Journal of Mathematical Problems in Engineering - Hidawi, 2015*(215147).

Industry Outlook. (2014). *Industry Outlook Data Center Energy Efficiency*. Retrieved October 10, 2018, from http://www.datacenterjournal.com/industry-outlook-data-center-energy-efficiency/

Intel. (2009). *The problem of power consumption in servers. .* Intel .

Intel. (2016). *Data Center Strategy Leading Intel's Business Transformation.* Intel.

Iqbal, R. A. (2012). Correlation aided Neural Networks: A correlation based approach of using feature importance to improve performance. *2012 International Conference on Informatics, Electronics & Vision (ICIEV).* Dhaka, Bangladesh: IEEE.

Ismael, S. M., Renyu, Y., Jie, X., & Tianyu, W. (2013). Improved Energy-Efficiency in Cloud Datacenters with Interference-Aware Virtual Machine Placement. *Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium* (pp. 1-8). Mexico City, Mexico: IEEE.

Jiankang, D., Hongbo, W., & Shiduan, C. (2015). *Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling.* China communications .

Jiaqi, Z., Yousri, M., Jie, T., Foued, J., Qinghuai, L., & Achim, S. (2014). Using a vision cognitive algorithm to schedule virtual machines. *International Journal of Applied Mathematics and Computer Science, 24*(3).

Jiaqing, D., Nipun, S., & Willy, Z. (2010). Performance profiling in a virtualized environment. *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. Boston, USA.

Joel, S. (2017). *Cloud Benchmarking: Estimating Cloud Application Performance Based on Micro Benchmark Profiling*. University of Zurich .

Jones, E. (2004). *An Introduction to Neural Networks.* San Ramon, CA: Visual Numerics, Inc.

Jungsoo, K., Martino, Ruggiero, D. A., & Marcel, L. (2013). Correlation-Aware Virtual Machine Allocation for Energy-Efficient Datacenters. *13 Proceedings of the Conference on Design, Automation and Test in Europe*, (pp. 1345-1350).

Jyoti, S., Shobha, G., & Anala, M. R. (2015). A State-of-art Comparison of Opensource IaaS Cloud Softwares. *American International Journal of Research in Science, Technology, Engineering & Mathematics*, 26-30.

Kamga, C. M. (2013). CPU frequency emulation based on DVFS. *2012 IEEE Fifth International Conference on Utility and Cloud Computing.* Chicago, IL, USA .

Karsoliya, S. (2012). Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture. *International Journal of Engineering Trends and Technology, 3*(6), 714-717.

Kenga, M. D., Omwenga, V., & Ogao, P. (2017). Energy Consumption in Cloud Computing Environments. *Pan African Conference on Science, Computing and Telecommunications (PACT) 2017.* Nairobi.

Khan, A., Paplinski, A., Khan, A. M., Murshed, M., & Buyya, R. (2018). Dynamic Virtual Machine Consolidation Algorithms for Energy-Efficient Cloud Resource Management: A Review. In *Sustainable Cloud and Energy Services.*

Khan, A., Yan, X., Tao, S., & Anerousis, i. (2012). Workload characterization and prediction in the cloud: A multiple time series approach. *Network Operations and Management Symposium (NOMS), 2012 IEEE.* IEEE.

Khashei, M., & Bijari, M. (2010). An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with Applications: Elsevier, 37*(1), 479-489.

Khosravi, A. (2017). *Energy and Carbon-Efficient Resource Management in Geographically Distributed Cloud Data Centers.* Melbourne, Australia: The University of Melbourne.

Kim, W., Gupta, M. S., Wei, G.-Y., & Brooks, D. (2018). System level analysis of fast, per-core DVFS using on-chip switching regulators. *2008 IEEE 14th International Symposium on High Performance Computer Architecture.* Salt Lake City, UT, USA.

Kivunja, C., & Kuyini, A. B. (2017). Understanding and Applying Research Paradigms in Educational Contexts. *International Journal of Higher Education, 6*(5), 26 - 41 .

Kumar, R. (2011). *Research Methodology: a step-by-step guide for beginners.* (Third, Ed.) London: Sage.

Kurpicz, M., Sobe, A., & Felber, P. (2014). Using power measurements as a basis for workload placement in heterogeneous multi-cloud environments. *Proceedings of the 2nd International Workshop on CrossCloud Systems.* New York, NY, USA .

Lidin, D., & Mohamed, S. (2014). Enhancing Minimal Virtual Machine Migration in Cloud Environment. *International Journal of Research in Engineering and Technology, 3*(7).

Live Consulting . (2016). *Virtual Servers Explained, Simply*. Retrieved from Live Consulting : https://www.liveconsulting.com/news/virtual-servers-explained-simply

Lu, Y., Panneerselvam, J., Liu, L., & Wu, Y. (2016). RVLBPNN: A Workload Forecasting Model for Smart Cloud Computing. *Scientific Programming: Hidawi, 2016*(5635673).

Mallavarapu, R. (2012). *Dynamic Resource Provisioning in IaaS Cloud Environment.* Aalto University.

Manoel, F., Oliveira, R., Monteiro, C., Inácio, P., & Freire, M. (2017). CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM).* Lisbon, Portugal.

Mar, C.-Z., Lavinia, S., Orgerie, A.-C., & Guillaume, P. (2016). *An experiment-driven energy consumption model for virtual machine management systems.* Research Report.

Maria, C. C., Marco, L. D., Luisa, M., Dana, P., Momin, I. M., & Tabash, D. T. (2016). *Workloads in the Clouds.*

Martin, A., & Marangozova-Martin, V. (2018). *Automatic benchmark profiling through advanced workflow-based trace analysis.* Wiley Online Library.

Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.-M., & Vasilakos, A. V. (2015). Cloud computing: survey on energy efficiency. *ACM Computing Surveys, 47*(2).

Melhem, S. B., Agarwal, A., Goel, N., & Zaman, M. (2018). Markov Prediction Model for Host Load Detection and VM Placement in Live Migration. *IEEE Access*, 7190 - 7205.

Meng, X., Isci, C., Kephart, J., Kephart, J., Bouillet, E., & Bouillet, E. (2010). Efficient resource provisioning in compute clouds via VM multiplexing. *Proceedings of the 7th international conference on Autonomic computing*, (pp. 11-20). Washington DC, USA.

Meng, X., Meng, X., Meng, X., Meng, X., Meng, X., & Meng, X. (2010). Efficient resource provisioning in compute clouds via VM multiplexing. *Proceedings of the 7th international conference on Autonomic computing* . Washington DC, USA .

Microsoft. (2016). *Project Natick*. Retrieved October 25, 2016, from Microsoft: http://natick.research.microsoft.com/

Microsoft. (2018). *Microsoft Azure*. Retrieved from Microsoft Azure: https://azure.microsoft.com/en-us/

Minet, P., Renault, E., Khoufi, I., & Boumerdassi, S. (2018). Analyzing Traces from a Google Data Center . *14th International Wireless Communications and Mobile Computing Conference*. Limassol, Cyprus: Hal.

Mirabel, A., & Siddiqui, R. (2015). *Energy Aware Consolidation in Cloud Computing.* California State University.

Mittal, R., Bajaj, S., & Neha. (2013). Leakage Power Reduction in CMOS. *International Journal of Engineering Research and Applications, 3*(3), 1365-1367.

Mohsen, S., Hadi, S., & Mahsa, N. (2011). Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques. *The Journal of Supercomputing, 61*(1).

Mozo, A., Ordozgoiti, B., & Gómez-Canaval, S. (2018). *Forecasting short-term data center network traffic load with convolutional neural networks.* PLOS one.

Natural Resources Defense Council (NRDC). (2014). *Data Center Efficiency Assessment.* NRDC. Retrieved from

https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf

Neeraj, M., Manpreet, S., & Sanjeev, K. R. (2016). Resource Scheduling in Cloud Environmet: a Survey. *Advances in Science and Technology, 10*(30).

Neha, R., & Rishabh, J. (2015). Cloud Computing: Architecture and Concept of Virtualization. *International Journal of Science, Technology & Management, 4*(1).

Nikravesh, A. Y., Ajila, S. A., & Lung, C.-H. (2017). An autonomic prediction suite for cloud resource provisioning. *Journal of Cloud Computing: Advances, Systems and Applications, 6*(3).

NIST. (2011). *The NIST Definition of Cloud Computing*. U.S. Department of Commerce. Retrieved from https://www.profsandhu.com/cs6393_s16/nist-SP800-145.pdf

O'Connor, G. (2014). *Moore's law gives way to Bezos's law*. Retrieved October 01, 2018, from Gigaom: https://gigaom.com/2014/04/19/moores-law-gives-way-to-bezoss-law/

Oxford Research. (2015). *Finland's Giant Data Center Opportunity* . Oxford Research.

ParkMyCloud. (2018). *Why Azure Right Sizing is Important* . (ParkMyCloud) Retrieved November 01, 2018, from https://www.parkmycloud.com/azure-right-sizing/

Patel, J., Jindal, V., Yen, I.-L., Bastani, F., Xu, J., & Garraghan, P. (2015). Workload Estimation for Improving Resource Management Decisions in the Cloud. *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems.* Taichung, Taiwan.

Patel, V., & Bheda, H. (2014). Reducing Energy Consumption with Dvfs for Real-Time Services in Cloud Computing. *IOSR Journal of Computer Engineering (IOSR-JCE), 16*(3), 53-57.

Perera, S. (2016). *Multi-tenancy after 10 years of Cloud Computing* . (Hackernoon) Retrieved November 01, 2018, from https://hackernoon.com/multi-tenancy-after-10-years-of-cloud-computing-19de782ef899

Phoronix Test Suite . (2018). *Phoronix Test Suite Suites*. (Phoronix Media) Retrieved August 05, 2018, from https://openbenchmarking.org/suites/pts

Phoronix Test Suite. (2018). *Phoromatic: Automated Linux Benchmark Management & Test Orchestration*. Retrieved from Phoronix Test Suite:

http://www.phoronix-test-suite.com/index.php?k=phoromatic#phoromatic

Phoronix Test Suite. (2018). *Phoronix Test Suite - Linux Testing and Benchmarking Platform, Automated Testing, Open-Source Benchmarking*. (Phoronix Media) Retrieved August 01, 2018, from https://www.phoronix-test-suite.com/

Prevost, J. J., Nagothu, K., Kelley, B., & Jamshidi, M. (2011). Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models . *Proceeding of the 2011 6th International Conference on System of Systems Engineering*. Albuquerque, New Mexico, USA.

Rais, I., Orgerie, A.-C., & Quinson, M. (2017). Impact of Shutdown Techniques for Energy-Efficient Cloud Data Centers. *16th International Conference on Algorithms and* . Granada, Spain .

Rallo, A. (2014). *Industry Outlook: Data Center Energy Efficiency*. Retrieved August 4, 2015, from Data Center Journal: http://www.datacenterjournal.com/industry-outlook-data-center-energy-efficiency/

Rasheduzzaman, M., Islam, M. A., Islam, T., Hossain, T., & Rahman, R. M. (2014). Task shape classification and workload characterization of google cluster trace . *Advance Computing Conference (IACC), 2014 IEEE International.* Gurgaon, India.

Reiss, C., & Wilkes, J. (2011). *Google cluster-usage traces: format + schema* . Google .

Ribeiro, C., Castro, M., Vania, M.-M., & Méhaut, J.-F. (2012). Evaluating CPU and Memory Affinity for Numerical Scientific Multithreaded Benchmarks on Multi-cores. *IADIS International Journal on Computer Science and Information Systems, 7*(1), 79-93.

Rice University . (2017). *RUBiS - Rice University Bidding System*. Retrieved from RUBiS : https://cs.nyu.edu/~totok/professional/software/rubis/rubis.html

Rodrigo, C., Rajiv, R., Anton, B., Cesar, D. R., & Rajkumar, B. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Journal of Software: Practise and Experience , 4*(1), 23-50.

Safaricom. (2019). *Safaricom*. Retrieved August 1, 2019, from CLOUD AND HOSTING SERVICES:

https://www.safaricom.co.ke/business/corporate/cloud-and-hosting-services

Sajitha, A., & Subhajini, A. (2018). Analysis of CloudSim Toolkit for Implementing Energy Efficient Green Cloud Data Centers. *nternational Journal for Research in Applied Science & Engineering Technology, 6*(6), 4614-4623.

Salam, I., Karim, R., & Ali, M. (2018). Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres. *Journal of Cloud ComputingAdvances, Systems and Applications, 7*(1).

Sampaio, A. M. (2015). *Energy-efficient and SLA-based Management of IaaS Cloud Data Centers.* University of Porto, Faculty of Engineering . University of Porto.

Sanjeev, V. (2015). Innovations in Technology: Cloud Computing and Energy Efficiency. *International Journal of Engineering and Management Sciences, 6*(2).

Sareh, F. P. (2016). *Energy-Efficient Management of Resources in Enterprise and Container-based Clouds.* The University of Melbourne , Department of Computing and Information Systems . The University of Melbourne .

Sareh, F. P., Calheiros, R. N., Chan, J., Dastjerdi, A. V., & Buyya, R. (2015). Virtual Machine Customization and Task Mapping Architecture for Efficient Allocation of Cloud Data Center Resources. *The Computer Journal*.

Scikit-learn. (2018). *Scikit-learn : Machine Learning in Python*. (Scikit-learn) Retrieved from https://scikit-learn.org/stable/index.html

Sharma, R. M. (2014). The Impact of Virtulization in Cloud Computing. *International Journal of Recent Development in Engineering and Technology, 3*(1).

Shen, S., Beek, V. v., & Iosup, A. (2015). Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters. *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. Shenzhen, China.

Shen, S., Beek, V. v., & Iosup, A. (2015). *Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters.* Delft University , Faculty of Information Technology and Systems Department . Parallel and Distributed Systems Section.

Shmueli, G., Nitin, R. P., Peter, C. B., Yahav, I., & Kenneth, C. L. (2017). Nerural nets. In *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner* (p. 271). Wiley.

Singh, R., Sharma, U., Cecchet, E., & Shenoy, P. (2010). Autonomic Mix-Aware Provisioning for Non-Stationary Data Center Workloads. *Proceedings of the 7th international conference on Autonomic computing*, (pp. 21-30).

Smith, J., & Sommerville, I. (2011). Workload Classification & Software Energy Measurement for Efficient Scheduling on Private Cloud Platforms. *Conference'10 University of St Andrews.*

Sondhi, A., Gupta, A., & Vivek, A. (2016). Power Savings in Green Cloud Environment Using K-Means Clustering. *International Journal of Scientific & Engineering Research, 7*(10), 1610 - 1614.

Song, W., Xiao, Z., Chen, Q., & Luo, H. (2014). Adaptive Resource Provisioning for the Cloud Using Online Bin Packing. *IEEE Transactions on Computers, 63*(11).

Speitkamp, B., & Bichler, M. (2010). A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. *IEEE Transactions on Services Computing , 3*(4), 266 - 278.

Subramaniam, B., & Feng, W.-c. (2013). Towards Energy-Proportional Computing Using Subsystem-Level Power Management. *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering.* Prague, Czech Republic. Retrieved from https://arxiv.org/pdf/1501.02724.pdf

Supriya, K., Rajesh, K., & Anju, S. (2014). Prediction Based Proactive Thermal Virtual Machine Scheduling in Green Clouds. *The Scientific World Journal*, 1-13.

Szefer, J. M. (2013). *Architectures for Secure Cloud Computing Servers.* Princeton University.

Team, T. (2014). *Intel Launches Its Most Energy-Efficient Processor To Spur PC Demand*. Retrieved January 26, 2017, from Forbes: http://www.forbes.com/sites/greatspeculations/2014/09/10/intel-launches-its-most-energy-efficient-processor-to-spur-pc-demand/#1c55373c7d07

Tesfatsion, S. K. (2018). *Energy-efficient cloud computing: Autonomic resource provisioning for datacenters.* Umea: Umea University .

TSO Logic. (2017). *TSO Logic*. Retrieved January 4, 2017, from TSO Logic : http://tsologic.com/

Ubuntu. (2018). *powerstat: A tool for measuring power consumption* . Retrieved from Ubuntu:

http://manpages.ubuntu.com/manpages/bionic/man8/powerstat.8.html

Ullah, Q. Z., Hassan, S., & Khan, G. M. (2017). Adaptive Resource Utilization Prediction System for Infrastructure as a Service Cloud. *Journal of Computational Intelligence and Neuroscience: Hidawi, 2017*(4873459).

Urul, G. ¨. (2018). *ENERGY EFFICIENT DYNAMIC VIRTUAL MACHINE ALLOCATION WITH CPU USAGE PREDICTION IN CLOUD DATACENTERS.* Bilkent University, Graduate School of Engineering and Science. Bilkent University.

Vasudevan, V., Andersen, D., Kaminsky, M., Tan, L., Franklin, J., & Moraru, I. (2010). Energy-efficient cluster computing with FAWN: workloads and implications. *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. Passau, Germany.

Verma, A., Gargi Dasgupta, T. K., Pradipta, D., & Ravi, K. (2009). Server Workload Analysis for Power Minimization using Consolidation. *in Proceedings of the 2009 Conference on USENIX Annual Technical Conference.* IBM India Research Lab.

Villebonnet, V., & Georges, D. C. (2014). Thermal-aware cloud middleware to reduce cooling needs . *IEEE International Conference on Collaboration Technologies and Infrastructures - WETICE.* Parma, Italy. .

VMware. (2015). *Performance Best Practices for VMware vSphere 6.0.* Palo Alto, CA: VMware, Inc.

VMware. (2015). *vSphere Resource Management.* Palo Alto, CA: VMware, Inc.

Vyom. (2013). *What is Cloud Computing?* Retrieved July 2, 2018, from Vyom.com: http://www.vyomtech.com/2013/10/30/what_is_cloud_computing.html

Waikato University . (2018). *Machine Learning at Waikato University* . (Waikato University) Retrieved November 25, 2018, from https://www.cs.waikato.ac.nz/ml/index.html

Wang, J., Cheng, C.-T., & Tse, C. (2016). Effects of Correlation-based VM Allocation Criteria to Cloud Data Center. *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (pp. 398-401). Chengdu, China: IEEE.

Xavier, M. G., Matteussi, K. J., Lorenzo, F., & Rose, C. A. (2016). Understanding performance interference in multi-tenant cloud databases and web applications. *2016 IEEE International Conference on Big Data* . Washington, DC, USA.

Xu, H., Zuo, X., Liu, C., & Zhao, X. (2016). Predicting Virtual Machine's Power via a RBF Neural Network. *International Conference in Swarm Intelligence.* Bali, Indonesia.

Xue, J., Yan, F., Birke, R., Chen, L., Scherer, T., & Smirni, E. (2015). PRACTISE: Robust prediction of data center time series. *2015 11th International Conference on Network and Service Management (CNSM).* Barcelona, Spain.

Xuesong, P., Barbara, P., & Monica, V. (2018). Virtual Machine Profiling for Analyzing Resource Usage of Applications. *International Conference on Services Computing* (pp. 103-118). Milano, Italy: Springer, Cham.

Yazdanov, L., & Fetzer, C. (2012). Vertical Scaling for Prioritized VMs Provisioning. *2012 Second International Conference on Cloud and Green Computing.* Xiangtan, China : IEEE.

Zhang, Q., Zhani, M. F., Zhang, S., Zhu, Q., Boutaba, R., & Hellerstein, J. L. (2012). Dynamic energy-aware capacity provisioning for cloud computing environments. *roceedings of the 9th ACM International Conference on Autonomic Computing* , (pp. 145-154).

Zhang, Y., & Ansari, N. (2013). Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation.

Zitao, L., & Sangyeun, C. (2012). Characterizing Machines and Workloads on a Google Cluster. *41st International Conference on Parallel Processing Workshops* (pp. 39 - 403). University of Pittsburgh.

Zoha, U., & Shailendra, S. (2015). A Survey of Virtual Machine Placement Techniques in a Cloud Data Center. *International Conference on Information Security & Privacy*, 491 – 498 .

Appendices

Appendix A

A.1 Java class for running simulation

```
public class MyPhDExample {
   private static final int HOSTS_8CORE = 26;
   private static final int HOSTS_12CORE = 23;
   private static final int HOST_PES_8cores = 8 ;
   private static final int HOST_PES_12cores = 12;
   private static final int SCHEDULING_INTERVAL = 2;

   private static final int VMS = 520;
   //private static final int VM_PES = 4; IS DYNAMIC

   private static final int CLOUDLETS = 520;
   public static final int WORKLOAD = 520;
   private static final long VM_MIPS_CAPACITY = 1000;
   private static final long VM_BW = 1000;
   private static final double BW_UTILIZATION = 0.5;

   private CloudSim simulation;
   private data centerBroker broker;
          private List<Vm> vmList;
   private List<Cloudlet> cloudletList;
   private data center data center0;
   private List<Host> hostList;
   public int vmCreationFailureCount = 0;

   /**
    * Defines the minimum percentage of power a Host uses,
    * even it it's idle.
    */
   private static final double STATIC_POWER_PERCENT = 0.6; //This is like 60% --use 60%

   /**
    * The max number of watt-second (Ws) of power a Host uses.
    */
```

```
private static final int MAX_POWER_WATTS_SEC = 100;
        /**
         * @param args
         */
private boolean showAllHostUtilizationHistoryEntries = true;

//getters and setters
public List<Vm> getVmList() {
                return vmList;
        }
/**
 * Set vm list
 * @param vmList
 */
        public void setVmList(List<Vm> vmList) {
                this.vmList = vmList;
        }

        public List<Cloudlet> getCloudletList() {
                return cloudletList;
        }
        /**
         * Set cloudlet list
         * @param cloudletList
         */
        public void setCloudletList(List<Cloudlet> cloudletList) {
                this.cloudletList = cloudletList;
        }


        public static void main(String[] args) {

                new MyPhDExample();
        }
//class constructor
public MyPhDExample() {
        System.out.println(getClass().getSimpleName() + " starting!");
        simulation = new CloudSim();
```

```
        data center0 = createdata centers imple();
        data center0.getCharacteristics().setVmm("VMware ESX");
        System.out.println(getClass().getSimpleName() + " 2!");
        //Creates a broker that is a software acting on behalf a cloud customer to manage his/her VMs and
Cloudlets
        broker = new data centerBrokerSimple(simulation);
        //System.out.println(getClass().getSimpleName() + " 3!");
        createVmsAndClouletLists(broker);
        //System.out.println(getClass().getSimpleName() + " 4!");
        vmList = getVmList();
        System.out.println(getClass().getSimpleName() + " 5!");
        cloudletList = getCloudletList();
        //System.out.println(getClass().getSimpleName() + " 6!");
        broker.submitVmList(vmList);
        //System.out.println(getClass().getSimpleName() + " 7!");
        broker.submitCloudletList(cloudletList);
        //System.out.println(getClass().getSimpleName() + " 8!");
        //start simulation
        simulation.start();
        printHostsCpuUtilizationAndPowerConsumption();
        System.out.println();
        System.out.println("Failed count: " + this.vmCreationFailureCount);

        System.out.println("vmm name" + data center0.getCharacteristics().getArchitecture());
            }

    public void onVmFailedToCreateOnHost(Vmdata centerEventInfo info) {
            this.vmCreationFailureCount++;
            }

    private void printHostsCpuUtilizationAndPowerConsumption() {
        System.out.println();
        for (final Host host : hostList) {
            printHostCpuUtilizationAndPowerConsumption(host);
        }
    }
```

```
    private void printHostCpuUtilizationAndPowerConsumption(final Host host) {
        System.out.printf("Host %d CPU utilization and power consumption\n", host.getId());
        final Map<Double, DoubleSummaryStatistics> utilizationPercentHistory = host.getUtilizationHistory();
        double totalPowerWattsSec = 0;
        double prevUtilizationPercent = -1, prevWattsPerInterval = -1;
        //time difference from the current to the previous line in the history
        double utilizationHistoryTimeInterval;
        double prevTime=0;
        for (Map.Entry<Double, DoubleSummaryStatistics> entry : utilizationPercentHistory.entrySet()) {
            utilizationHistoryTimeInterval = entry.getKey() - prevTime;
            //The total Host's CPU utilization for the time specified by the map key
            final double utilizationPercent = entry.getValue().getSum();
            final double wattsSec = host.getPowerModel().getPower(utilizationPercent);
            final double wattsPerInterval = wattsSec*utilizationHistoryTimeInterval;
            totalPowerWattsSec += wattsPerInterval;
            //only prints when the next utilization is different from the previous one, or it's the first one
            if(showAllHostUtilizationHistoryEntries || prevUtilizationPercent != utilizationPercent ||
prevWattsPerInterval != wattsPerInterval) {
                System.out.printf("\tTime %8.2f | CPU Utilization %6.2f%% | Power Consumption: %8.0f Watt-
Sec * %.0f Secs = %.0f Watt-Sec\n",
                    entry.getKey(), utilizationPercent * 100, wattsSec, utilizationHistoryTimeInterval,
wattsPerInterval);
            }
            prevUtilizationPercent = utilizationPercent;
            prevWattsPerInterval = wattsPerInterval;
            prevTime = entry.getKey();
        }

        System.out.printf(
            "Total Host %d Power Consumption in %.0f secs: %.0f Watt-Sec (%.5f KWatt-Hour)\n",
            host.getId(), simulation.clock(), totalPowerWattsSec,
PowerAware.wattsSecToKWattsHour(totalPowerWattsSec));
        final double powerWattsSecMean = totalPowerWattsSec / simulation.clock();
        System.out.printf(
            "Mean %.2f Watt-Sec for %d usage samples (%.5f KWatt-Hour)\n",
            powerWattsSecMean, utilizationPercentHistory.size(),
PowerAware.wattsSecToKWattsHour(powerWattsSecMean));
```

```
}
        //creating a data center
        private data center createdata centers imple() {
                hostList = new ArrayList<>(HOSTS_12CORE + HOSTS_8CORE);

                int count = 1;
                for(int i = 0; i < HOSTS_12CORE; i++) {

                        Host host = create12CorePowerHost(count);

      hostList.add(host);
      //System.out.println("host No. " + count);
      count++;
    }


    for(int i = HOSTS_12CORE; i < HOSTS_8CORE + HOSTS_12CORE; i++) {
      Host host = create8CorePowerHost(count);
      hostList.add(host);
      //System.out.println("host No. " + count);
      count++;
    }
    //VmAllocationPolicySimple is same as Worst Fit (WF)
    //final data center dc = new data centers imple(simulation, hostList, new
VmAllocationPolicySimple());
    final data center dc = new data centers imple(simulation, hostList, new
VMAllocationPolicyFirstFitIncreasingSimilarity());
    dc.setSchedulingInterval(SCHEDULING_INTERVAL);
    return dc;
        }
        //creating hosts to be put in data center
        private Host create8CorePowerHost(long hostId) {
                List<Pe> peList = new ArrayList<>(HOST_PES_8cores);
                //List of Host's CPUs (Processing Elements, PEs)
    for (int i = 0; i < HOST_PES_8cores; i++) { //Adding CORES to the host machine
      peList.add(new PeSimple(1000, new PeProvisionerSimple()));// 1000 is the PE/core processing
capacity in MIPS
    }
```

```
        final long ram = 148000; //in Megabytes
        final long bw = 10000; //in Megabits/s//I have no reason to change this
        final long storage = 3000000; //in Megabytes

        final Host host = new HostSimple(ram, bw, storage, peList);
        host.setPowerModel(new PowerModelLinear(MAX_POWER_WATTS_SEC,
STATIC_POWER_PERCENT));//power model for computing power consumption.
        host
          .setRamProvisioner(new ResourceProvisionerSimple())
          .setBwProvisioner(new ResourceProvisionerSimple())
          .setVmScheduler(new VmSchedulerTimeShared())
          .setId(hostId);;
        return host;
            }

            private Host create12CorePowerHost(long hostID) {
                    List<Pe> peList = new ArrayList<>(HOST_PES_12cores);
                    //List of Host's CPUs (Processing Elements, PEs)
        for (int i = 0; i < HOST_PES_12cores; i++) { //Adding CORES to the host machine
          peList.add(new PeSimple(1000, new PeProvisionerSimple()));// 1000 is the PE/core processing
capacity in MIPS
        }
        final long ram = 148000; //in Megabytes
        final long bw = 10000; //in Megabits/s//I have no reason to change this
        final long storage = 3000000; //in Megabytes

        final Host host = new HostSimple(ram, bw, storage, peList);
        host.setPowerModel(new PowerModelLinear(MAX_POWER_WATTS_SEC,
STATIC_POWER_PERCENT));//power model for computing power consumption.
        host
          .setRamProvisioner(new ResourceProvisionerSimple())
          .setBwProvisioner(new ResourceProvisionerSimple())
          .setVmScheduler(new VmSchedulerTimeShared())
          .setId(hostID);;
        return host;

            }
```

```
//creating a list of VMs
public void createVmsAndClouletLists(data centerBroker broker) {
        final List<Vm> vmlist = new ArrayList<>(VMS);
        final List<Cloudlet> cloudletlist = new ArrayList<>(CLOUDLETS);
        final List<String[]> workloadlist = new CSVFileReader().processCsv();

        ;
        //System.out.println(workloadlist.size());
        //for(int i = 0; i<520; i++) {
        for(int i = 0; i<workloadlist.size(); i++) {
                //create a VM

                String[] workloadRow = workloadlist.get(i);

                Vm vm = new VmSimple(Integer.parseInt(workloadRow[0].trim()),
VM_MIPS_CAPACITY,Long.parseLong(workloadRow[2].trim()));

                vm.setRam(Math.round(Double.parseDouble(workloadRow[4].trim())/1000))
                .setBw(VM_BW)
                .setSize(Math.round(Double.parseDouble(workloadRow[7].trim())*1000))
                .setCloudletScheduler(new CloudletSchedulerTimeShared())
                .setDescription(workloadRow[1]);
                vm.getUtilizationHistory().enable();
                vm.addOnCreationFailureListener(this::onVmFailedToCreateOnHost);
                //add this vm to a vmlist list

                vmlist.add(vm);
                setVmList(vmlist);
                //create a cloudlet utilization models for RAM, CPU and Bandwidth.
                //final UtilizationModel cloudletUtilizationModelCpu = new
UtilizationModelDynamic(Math.ceil(Double.parseDouble(workloadRow[3].trim())))/100);
                //final UtilizationModel cloudletUtilizationModelRam = new
UtilizationModelDynamic(Math.ceil(Double.parseDouble(workloadRow[6].trim())))/100);

                final UtilizationModel cloudletUtilizationModelCpu = new
UtilizationModelDynamic(0.8);
                final UtilizationModel cloudletUtilizationModelRam = new
UtilizationModelDynamic(0.8);
```

```
                    final UtilizationModel cloudletUtilizationModelBw = new
UtilizationModelDynamic(0.8);
                    //Now create a cloudlet
                final Cloudlet cloudlet =
                    new
CloudletSimple(Long.parseLong(workloadRow[0]),(long)Math.ceil(Double.parseDouble(workloadRow[5].trim
())/100),vm.getNumberOfPes())
                        .setFileSize((long)Math.ceil(Double.parseDouble(workloadRow[9].trim())*10))
                        .setOutputSize((long)Math.ceil(Double.parseDouble(workloadRow[9].trim())*10))
                        .setUtilizationModelCpu(cloudletUtilizationModelCpu)
                        .setUtilizationModelRam(cloudletUtilizationModelRam)
                        .setUtilizationModelBw(cloudletUtilizationModelBw);
                //add this cloudlet to a vmlist cloudletlist
                cloudletlist.add(cloudlet);
                setCloudletList(cloudletlist);
                //now bind this cloudlet to this vm
                broker.bindCloudletToVm(cloudlet, vm);
                //cloudlet.setVm(vm);
                    }

            }
}
```

A.2 Proposed VM allocation algorithms java class

```
public class VMAllocationPolicyFirstFitIncreasingSimilarity extends VmAllocationPolicyAbstract {
	@Override
	public Optional<Host> findHostForVm(Vm vm) {

		//get VM group from its decription.
		String vm_group = vm.getDescription().trim();
		//fetch all hosts
		List<Host> allAvailableHost = getHostList();
		//create a new candidate hosts (those with have enough resources to host the VM)
		List<Host> candidateHosts = new ArrayList<>();
		for(int i=0; i<allAvailableHost.size(); i++) {
			if(allAvailableHost.get(i).isSuitableForVm(vm)){
				candidateHosts.add(allAvailableHost.get(i));
			}
		}
		//In the candidateHosts, we compute similarity index with incoming VM for all hosts
and then order them in ascending order.
		Map<Host, Double> hostSimilarityIndexMap = new HashMap<>();
		for(int i=0; i<candidateHosts.size(); i++) {
			Host currentHost = candidateHosts.get(i);
			int totalNumberOfVmsAllocated = currentHost.getVmList().size();
			List<Vm> currentVmList =  currentHost.getVmCreatedList();
			//Get all similar vms
			int similarVms = 0;
			for(int j=0; j<currentVmList.size(); j++) {
				if(currentVmList.get(j).getDescription().trim().equals(vm_group)) {
					similarVms++;
				}
			}

			double similarityIndex = 0; //similarVms/totalNumberOfVmsAllocated;
			if(totalNumberOfVmsAllocated != 0) {
				similarityIndex = similarVms/totalNumberOfVmsAllocated;
			}
			hostSimilarityIndexMap.put(currentHost, similarityIndex);
		}
```

```
            return hostSimilarityIndexMap
                                .entrySet()
                    .stream()
                    .min(Comparator.comparingDouble(Map.Entry::getValue))
                    .map(Map.Entry::getKey);
        }

}
```

Appendix B

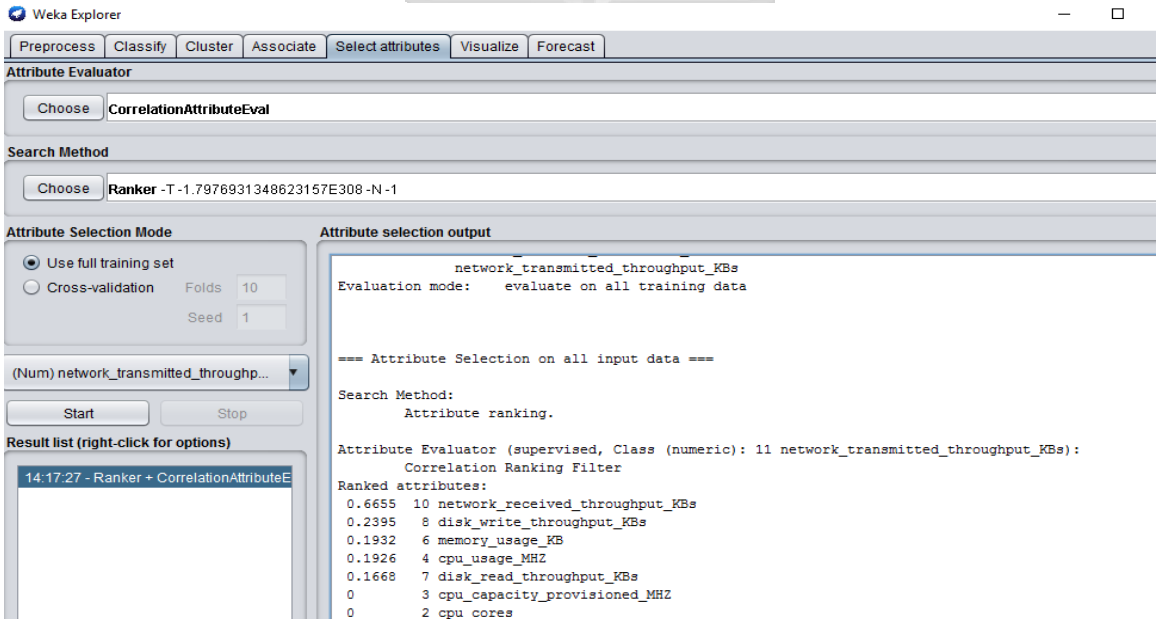
B.1 Sample WEKA ARFF file format

```
@RELATION vm_45

@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"
@ATTRIBUTE cpu_cores  NUMERIC
@ATTRIBUTE cpu_capacity_provisioned_MHZ NUMERIC
@ATTRIBUTE cpu_usage_MHZ NUMERIC
@ATTRIBUTE memory_capacity_provisioned_KB NUMERIC
@ATTRIBUTE memory_usage_KB NUMERIC
@ATTRIBUTE disk_read_throughput_KBs NUMERIC
@ATTRIBUTE disk_write_throughput_KBs NUMERIC
@ATTRIBUTE disk_size_GB NUMERIC
@ATTRIBUTE network_received_throughput_KBs NUMERIC
@ATTRIBUTE network_transmitted_throughput_KBs NUMERIC

@DATA
'2015-11-05 00:00:00',2,0,277,12582912,1424386,8,79,208,9,10
'2015-11-05 00:05:00',2,0,294,12582912,1667236,84,118,208,11,12
'2015-11-05 00:10:00',2,0,275,12582912,1374054,101,81,208,9,10
'2015-11-05 00:15:00',2,0,275,12582912,1357696,13,83,208,9,10
'2015-11-05 00:20:00',2,0,826,12582912,6146752,133108,523,208,18,294
'2015-11-05 00:25:00',2,0,269,12582912,4780248,12,74,208,9,10
'2015-11-05 00:30:00',2,0,921,12582912,4336071,23496,1036,208,160,6334
'2015-11-05 00:35:00',2,0,268,12582912,4327263,66,77,208,9,10
'2015-11-05 00:40:00',2,0,262,12582912,1861013,8,74,208,9,10
'2015-11-05 00:45:00',2,0,258,12582912,980209,7,76,208,9,10
'2015-11-05 00:50:00',2,0,513,12582912,1810681,958,1392,208,19,560
```

B.2 WEKA feature reduction example on VM 45



Appendix C

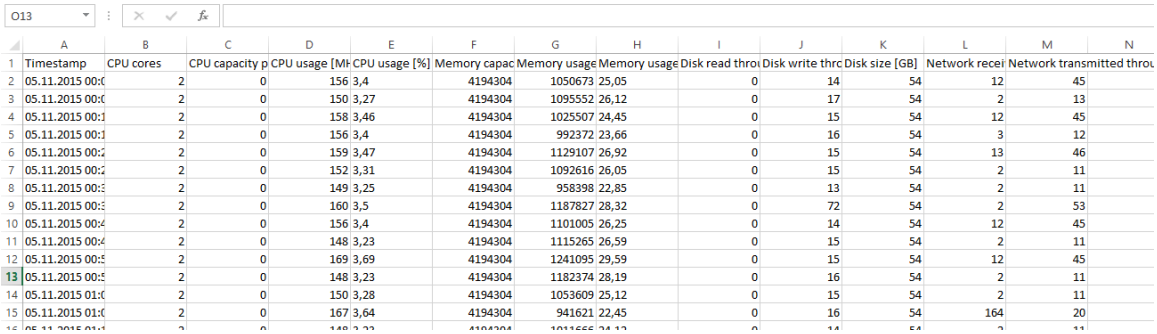
Reading power usage using the powerstat command

```
dkenga@fit33:~$ powerstat -tfR 0.5
Running for 60.0 seconds (120 samples at 0.5 second intervals).
Power measurements will start in 0 seconds time.

  Time     User  Nice   Sys  Idle    IO  Run Ctxt/s  IRQ/s  Watts  acpitz acpitz x86_pk  CPU Freq
14:28:40  98.2   0.0   1.5   0.2   0.0   10   6036   3646  51.32   29.80  27.80  62.00  3.79 GHz
14:28:41  99.2   0.0   0.8   0.0   0.0   10   7162   3266  51.25   29.80  27.80  62.00  3.79 GHz
14:28:41  99.0   0.0   0.8   0.2   0.0    9   5728   3786  51.09   29.80  27.80  60.00  3.79 GHz
14:28:42  99.2   0.0   0.8   0.0   0.0   10   6130   3566  51.03   29.80  27.80  63.00  3.79 GHz
14:28:42  98.8   0.0   0.5   0.8   0.0    8   5792   3376  50.94   29.80  27.80  63.00  3.79 GHz
14:28:43  98.2   0.0   0.8   1.0   0.0   10   5536   3626  50.21   29.80  27.80  63.00  3.79 GHz
14:28:43  83.1   0.0   1.3  15.7   0.0    7   6414   3032  47.89   29.80  27.80  58.00  3.80 GHz
14:28:44  82.5   0.0   1.5  16.0   0.0    9   2364   2060  52.01   29.80  27.80  64.00  3.79 GHz
14:28:44  99.0   0.0   1.0   0.0   0.0    9    508   2218  56.02   29.80  27.80  65.00  3.79 GHz
14:28:45  99.8   0.0   0.2   0.0   0.0    9    552   2246  56.19   29.80  27.80  65.00  3.79 GHz
14:28:45  99.5   0.0   0.5   0.0   0.0    9    518   2266  56.20   29.80  27.80  64.00  3.79 GHz
14:28:46 100.0   0.0   0.0   0.0   0.0    9    460   2214  56.19   29.80  27.80  65.00  3.79 GHz
14:28:46 100.0   0.0   0.0   0.0   0.0    9    474   2244  56.20   29.80  27.80  65.00  3.79 GHz
14:28:47 100.0   0.0   0.0   0.0   0.0    9    542   2242  55.88   29.80  27.80  65.00  3.79 GHz
14:28:47 100.0   0.0   0.0   0.0   0.0    9    540   2248  56.32   29.80  27.80  65.00  3.79 GHz
14:28:48 100.0   0.0   0.0   0.0   0.0    9    452   2218  56.07   29.80  27.80  67.00  3.79 GHz
14:28:48  99.8   0.0   0.2   0.0   0.0    9    512   2234  56.24   29.80  27.80  66.00  3.79 GHz
14:28:49 100.0   0.0   0.0   0.0   0.0    9    552   2248  55.81   29.80  27.80  66.00  3.79 GHz
```

Appendix D

Sample GWA-T-13 workload csv file

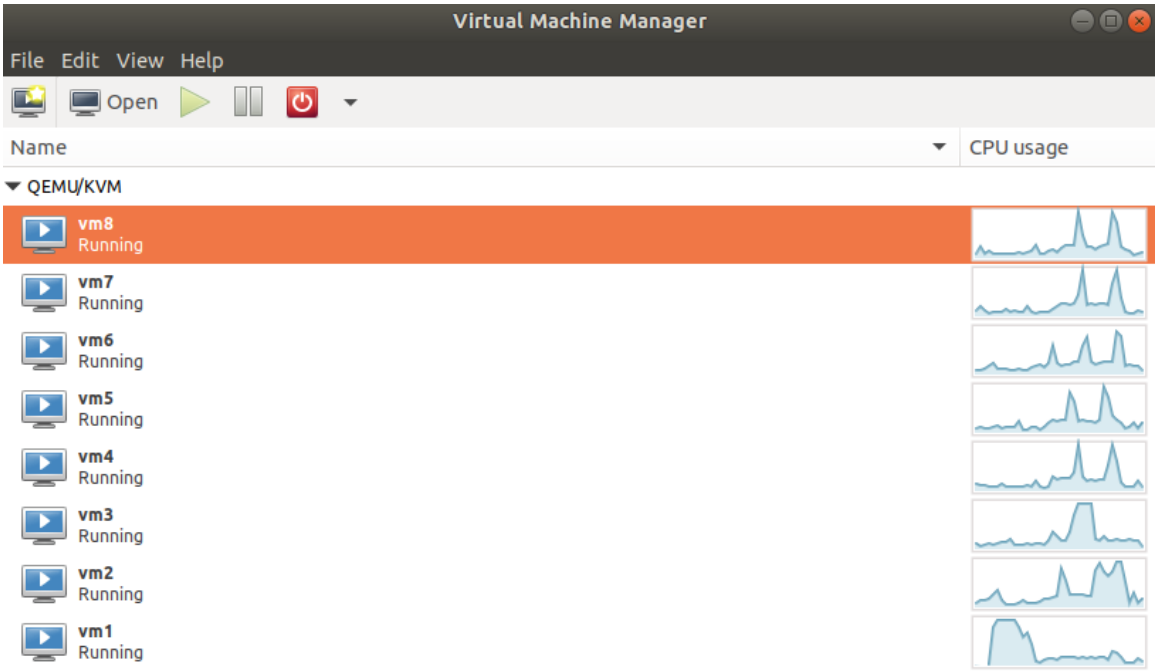


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Timestamp	CPU cores	CPU capacity p	CPU usage [MH	CPU usage [%]	Memory capac	Memory usage	Memory usage	Disk read throu	Disk write thro	Disk size [GB]	Network recei	Network transmitted throu	
2	05.11.2015 00:(	2	0	156	3,4	4194304	1050673	25,05	0	14	54	12	45	
3	05.11.2015 00:(	2	0	150	3,27	4194304	1095552	26,12	0	17	54	2	13	
4	05.11.2015 00:1	2	0	158	3,46	4194304	1025507	24,45	0	15	54	12	45	
5	05.11.2015 00:1	2	0	156	3,4	4194304	992372	23,66	0	16	54	3	12	
6	05.11.2015 00:2	2	0	159	3,47	4194304	1129107	26,92	0	15	54	13	46	
7	05.11.2015 00:2	2	0	152	3,31	4194304	1092616	26,05	0	15	54	2	11	
8	05.11.2015 00:3	2	0	149	3,25	4194304	958398	22,85	0	13	54	2	11	
9	05.11.2015 00:3	2	0	160	3,5	4194304	1187827	28,32	0	72	54	2	53	
10	05.11.2015 00:4	2	0	156	3,4	4194304	1101005	26,25	0	14	54	12	45	
11	05.11.2015 00:4	2	0	148	3,23	4194304	1115265	26,59	0	15	54	2	11	
12	05.11.2015 00:5	2	0	169	3,69	4194304	1241095	29,59	0	15	54	12	45	
13	05.11.2015 00:5	2	0	148	3,23	4194304	1182374	28,19	0	16	54	2	11	
14	05.11.2015 01:(	2	0	150	3,28	4194304	1053609	25,12	0	15	54	2	11	
15	05.11.2015 01:(	2	0	167	3,64	4194304	941621	22,45	0	16	54	164	20	

*0 indicates a missing value

Appendix E

Screenshot showing VMM with the 8 VMs



Appendix F

F.1 Starting Phoromatic server

```
dkenga@fit33:~$ phoronix-test-suite start-phoromatic-server

Port 8197 chosen as random port for this instance. Change the default port via the Phoronix Test Suite user configuration
Phoronix Test Suite v8.2.0 (Rakkestad) starting Phoromatic Server
Phoronix Test Suite User-Data Directory Path: /home/dkenga/.phoronix-test-suite/
Phoronix Test Suite Configuration File: /home/dkenga/.phoronix-test-suite/user-config.xml
Phoromatic Server Log File: /home/dkenga/.phoronix-test-suite/phoromatic.log

Launching with PHP built-in web server.



WebSocket Server Active: localhost:8541
The Phoromatic Web Interface Is Accessible At: http://localhost:8197
Press [ENTER] to kill server...
```

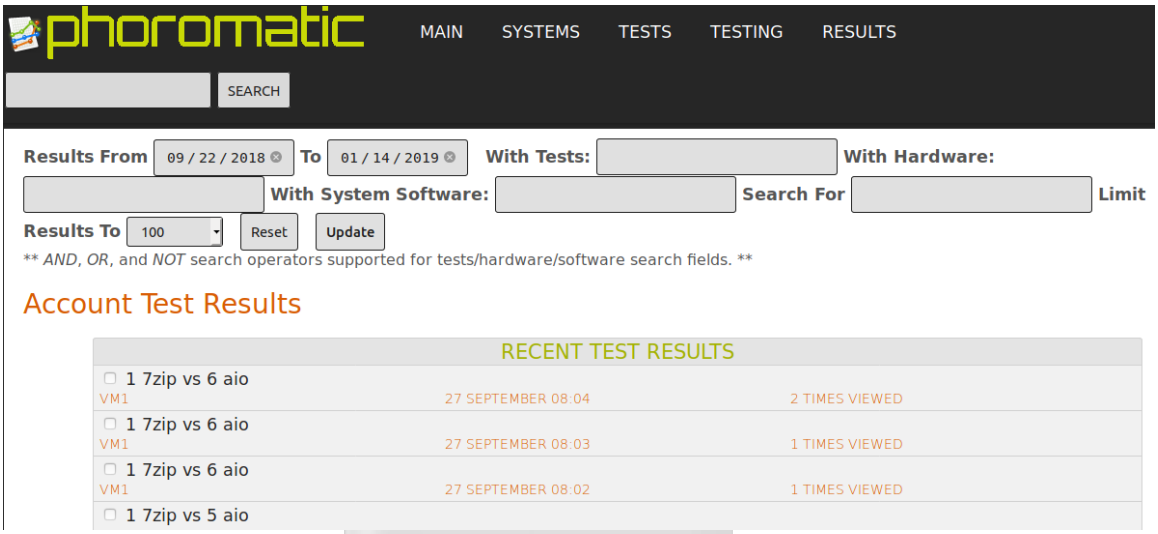
F.2 Phoromatic server web interface monitoring VMs, which workloads

Systems

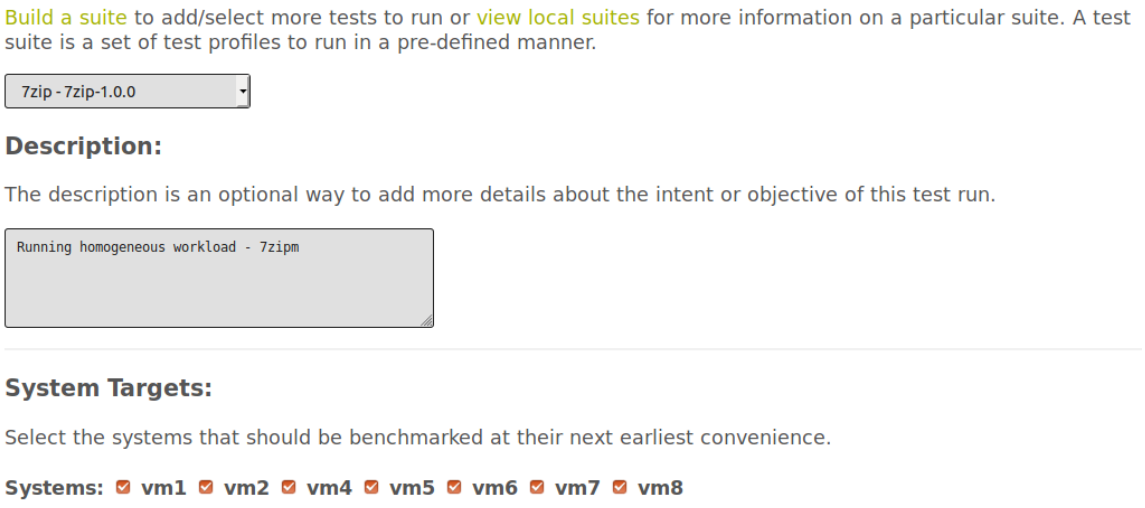
Various system interaction vitals for the Phoronix Test Suite systems associated with this account.

	Last Communication	Current Task	Phoronix Test Suite	Last IP	MAC	Wake-On-LAN	Latest Result Upload
vm1	09:03 27 September	Unknown	8.2.0 [8200]	192.168.122.241	52:54:00:5a:31:9a	N/A - Blocked	27 September
vm2	07:50 27 September	Unknown	8.2.0 [8200]	192.168.122.174	52:54:00:ed:58:42	N/A - Permitted	26 September
vm4	19:26 26 September	Unknown	8.2.0 [8200]	192.168.122.162	52:54:00:89:bf:e3	N/A - Permitted	26 September
vm8	17:51 26 September	Unknown	8.2.0 [8200]	192.168.122.157	52:54:00:74:1b:1e	N/A - Permitted	26 September
vm7	16:20 26 September	Unknown	8.2.0 [8200]	192.168.122.244	52:54:00:13:0e:a4	N/A - Permitted	26 September
vm6	15:08 26 September	Unknown	8.2.0 [8200]	192.168.122.67	52:54:00:8b:e9:3a	N/A - Permitted	26 September
vm5	15:07 26 September	Unknown	8.2.0 [8200]	192.168.122.151	52:54:00:ce:7d:e6	N/A - Permitted	26 September

F.3 Phoromatic server workload execution results from web interface



F.4 Phoromatic server showing how to run workload benchmarks



Appendix F

Turnitin thesis originality report