PERFORMANCE MODELING OF VIRTUAL MACHINES:

A Case Study of University of Agriculture, Makurdi Datacenter

By

EGBUNU, ISAAC BANYA

DEPARTMENT OF MATHEMATICS

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA.

JANUARY, 2015

PERFORMANCE MODELING OF VIRTUAL MACHINES:

A Case Study of University of Agriculture, Makurdi Datacenter

By

Isaac Banya EGBUNU, B.Sc Stat./Computer Science (FUAM) 2007

Msc/Scie/09163/2010-2011

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE

AWARD OF A

MASTER DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS

FACULTY OF SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA.

JANUARY, 2015

DECLARATION

I declare that the work in this thesis entitled "Performance Modeling of Virtual Machines (Case Study of University of Agriculture MakurdiDataCentre)" has been carried out by me in the Department of Mathematics under the supervision of Dr. S.E. Abdullahi and Dr. M. B. Hammawa. The information derived from the literature has been duly acknowledged in the text and the list of references is provided. No part of this thesis was previously presented for another degree or diploma at any other Institution.

Isaac, BanyaEgbunu _____ _____

Name of student Signature Date

CERTIFICATION

This thesis entitled “PERFORMANCE MODELING OF VIRTUAL MACHINES (Case Study of University of Agriculture MakurdiDataCentre)” by ISAAC BANYA EGBUNU meets the regulation governing the award of the degree of M.Sc Computer Science of Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

_____ _____

Chairman, Supervisory Committee Date

Dr. S.E. Abdullahi

_____ _____

Member, Supervisory Committee Date

Dr. M. B. Hammawa

_____ _____

External Examiner Date

_____ _____

H.O.D. Mathematics Date

Prof. BabangidaSani

_____ _____

Dean, School of Postgraduate Studies Date

Prof. Hassan Zoaka

ACKNOWLEDGEMENT

I sincerely appreciate the grace and mercy of God that sustained me throughout the programme.

I appreciate my supervisor Dr. S.E Abdullahi for his guide, support, and benevolence. I also appreciate my minor supervisor Dr. M. B. Hammawa for his understanding and support.

I thank the Director of ICT, University of Agriculture Makurdi, Eng. Dr. NentaweGoshwe for granting me unrestricted access to ICT infrastructures that has made my research work smooth.

I thank my family members and friends for their support and prayers that ensured the success of the programme.

ABSTRACT

With active deployment of virtualization in the University of Agriculture, Makurdi datacenter, allocation and scheduling of virtual CPU raises more challenges and have impacts on system performance due to workload fluctuations and changing needs. The objectives were to deploy a typical datacenter using VMware hypervisor and improve the reliability of Zhukui et al (2009) performance model by considering number of cores available to Virtual machine as well as custom workload on production environment.In this work, the researcher presented a model to predict the performance of consolidated virtual machines hosted on VMware Hypervisor 5.1 using Ubuntu 12.1 and windows server 2008 as guest operating system. Using CPU controlled approach for 20%, 40%, 60%, 80% and 100% allocations and workload generatorson testbeds and production environments, data was collected at peak periods during student registration processesusing basic system performance matrix on the virtual machines at intervals of 60 seconds and analyzed using SPSS package.Empirical result showed that proposed modelpredicted values are closer to the observed values than the Zhukui et' al (2009) for the three(3) workloads considered. Thus by varying the CPU allocation based on business needs, the model predicted the response time as observed in the production environment.

TABLE OF CONTENTS

Declaration iv

Certification v

Acknowledgement vi

Abstract vii

List of Figures x

List of Tables xi

List of Appendices xii

List of Abreviations xiii

CHAPTER ONE: INTRODUCTION

1.1 Background to the Research 1

1.2 Problem Statement 2

1.3 Justification for Case Study 3

1.4Significance of Study 4

1.5 Research Questions 5

1.6Aim and Objectives 6

1.7 Research Methodology 7

CHAPTER TWO: BACKGROUND AND RELATED WORK

2.1 Introduction 8

2.2 Data Centers and its Challenges 8

2.3 Server Virtualization 11

2.4 Virtualization in Datacenters 11

2.5 Understanding VMware Hypervisors 12

2.6 Evaluating System Performance in Datacenters .12

2.7 Theories on System Performance 15

2.7.1 Queuing Theory……..……..……..……..15

2.7.2 Use of Percentiles…….……..……..……..……..16

2.7.3 Statistical based Models……..……..……..16

2.8 Research Model …17

CHAPTER THREE: MATERIALS AND METHODS

3.1 Introduction 19

3.2 Assumptions 19

3.3 Experimental Setup 21

3.3.1Experimental Setup using Standard RUBiS Workload……..21

3.3.2Experimental Setup using Custom Workload…….22

3.4 Data Collection 26

3.5Model Generation 26

CHAPTER FOUR: RESULT AND DISCUSSION

4.1 Introduction 30

4.2 Model Evaluation based on Standard RUBiS Workload 30

4.3 Model Evaluation based on ASP.NET/IIS Workload 35

4.4 Model Evaluation based on Apache/PHP Workload 39

4.5 Answers to Research Question 45

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Introduction 47

5.2 Summary 47

5.3 Conclusion 49

5.4Recommendations 49

References 50

Appendix A: Performance Matrics Of RUBiS Workload 58

Appendix B: Performance Matrics Of exMaster Workload 61

Appendix C: Performance Matrics Of AppSton Workload 63

Appendix D: Comparison of base Model and the Proposed Model 65

LIST OF FIGURES

Fig 3.0 Experimental setup in virtualized environment for standard workload 22

Fig 3.1 Experimental setup in virtualized environment for custom workload 25

Fig 4.1 Performance prediction of VM Mean Resp Time for CPU allocation at 100% (RUBiS) 32

Fig 4.2 Performance prediction of VM Mean Resp Time for CPU allocation at 80% (RUBiS) 32

Fig 4.3 Performance prediction of VM Mean Resp Time for CPU allocation at 60% (RUBiS) 33

Fig 4.4 Performance prediction of VM Mean Resp Time for CPU allocation at 40% (RUBiS) 33
Fig 4.5 Performance prediction of VM Mean Resp Time for CPU allocation at 20% (RUBiS) 34
Fig 4.6 Performance prediction of VM Mean Resp Time for CPU allocation at 100% (exMaster) 36
Fig 4.7 Performance prediction of VM Mean Resp Time for CPU allocation at 80% (exMaster) 36
Fig 4.8 Performance prediction of VM Mean Resp Time for CPU allocation at 60% (exMaster) 37
Fig 4.9 Performance prediction of VM Mean Resp Time for CPU allocation at 40% (exMaster) 37
Fig 4.10 Performance prediction of VM Mean Resp Time for CPU allocation at 20% (exMaster) 38
Fig 4.11 Performance prediction of VM Mean Resp Time for CPU allocation at 100% (AppSton) 40
Fig 4.12 Performance prediction of VM Mean Resp Time for CPU allocation at 80% (AppSton) 40
Fig 4.13 Performance prediction of VM Mean Resp Time for CPU allocation at 60% (AppSton) 41
Fig 4.14 Performance prediction of VM Mean Resp Time for CPU allocation at 40% (AppSton) 41
Fig 4.15 Performance prediction of VM Mean Resp Time for CPU allocation at 20% (AppSton) 42

LIST OF TABLES

Table 2.1 Summary of previous work in virtualization 9

Table 2.2Summary of related work in performance modeling of virtual machines 18

Table 3.1IIS process worker log description 23

Table 4.1 Sample linear regression analysis for RUBiS workload 31

Table 4.2 Mean response time estimation error for RUBiS experiment 34

Table 4.3 Sample linear regression analysis for exMaster workload 35

Table 4.4 Mean response time estimation error for ASP.Net/IIS experiment 38

Table 4.6 Sample linear regression analysis for AppSton workload 39

Table 4.7 Mean response time estimation error for Apache/PHP experiment 43

Table 5.1Comparison of base model and proposed model 47

LIST OF APPENDICES

Appendix A: Performance Matrix for Virtual Machines based on Standard workload (RUBiS).….60

Appendix B: Performance Matrix for Virtual Machines based on Custom workload (exMaster)….63

Appendix C: Performance Matrix for Virtual Machines based on custom workload (AppSton)..….65

Appendix D: Comparion of base Model and Proposed Model ……..67

LIST OF ABBREVIATIONS

DRS – Dynamic resource scheduler

ICT – Information and Communication Technology

LQR – Linear Quadratic Regulator

LUN – Logical Unit Number

OS – Operating System

PHP- Personal Home Page

QOS- Quality of services

RAID-Random Access Independent Disk

RUBiS – Rice University Billing System

SAN – Storage Area Network

SOL- Service level Objectives

VM- Virtual Machines

Chapter One

INTRODUCTION

1.1BACKGROUND

Internet and business applications are increasingly being moved to large data centers that hold massive server and storage clusters. Currently data centers can even contain tens or hundreds of thousands of servers, and plans are already being made for data centers holding over a million servers (Appleby *et al.*, 2001). Some data centers are built to run applications for a 'single' company, such as the search engine clusters run by Google. Other data centers are operated by service providers that are able to rent storage and computation resources to other customers at very low cost due to their large scale.

Virtualization in an IT environment is essentially the isolation of one computing resource from others. By separating the different layers in the logic stack, you enable greater flexibility and simplify change management. Although services on dedicated servers achieve performance isolation, but it also leads to lower resource utilization, higher operational costs and power waste. Thus there is a strong incentive for consolidating resources among hardware and services, which is the so-called virtualization technology (Cong-Feng *et al.*, 2011).

A recent focus on reducing the economic costs of information technology (IT) motivates increased resource sharing and on-demand computing. Towards this, virtualization technologies enable IT resources to be dynamically allocated among multiple applications. Such a model empowers organizations to flex their computing resources based on workloads and business needs, and hence improve the efficiency of IT operations (Kephart *et al.*, 2007).

According to (Rolia *et al*., 2005), Virtualization can be used to "slice" a single physical host into one or more virtual machines (VMs) that share its resources. This can be useful in a hosting environment where customers or applications do not need the full power of a single server. In such a case, virtualization provides an easy way to isolate and partition server resources. The abstraction layer between the VM and its physical host also allows for greater control over resource management (Heo *et al*., 2007).

As data centers continue to deploy virtualized services, new problems have emerged such as determining optimal VM placements and dealing with virtualization overheads. At the same time, virtualization allows for new and better solutions to existing data center problems of power and maintenance cost of physical servers by allowing for rapid, flexible resource provisioning.

1.2PROBLEM STATEMENT

Resource contention is intensified in virtualized environments due to the consolidation nature of virtualization. Hence, the problem is how tominimize the allocation of CPU server resource to an IT service (or application), while satisfying Service Level Objectives (SLOs).

This work is an improvement on the approach considered by Zhikui*et al.* (2009)in modelling application performance in virtual machine. Zhikui*et al.(2009)*proposed themodel

$$T = \frac{1}{\lambda} \sum_{m=1}^{M} \frac{C_m}{e_m - C_m}$$

Where

T is the response time, λ is the workload, C_m is the consumption in tier m, e_m is the allocated CPU resource in tier m. The model predicts resourcedemand to meet application-level

performance requirementbased on workload transaction-mix history. However, the following problems were identified.

i. Approach failed to maximize server resources through consolidation
ii. Approach failed to consider other hypervisors (Vmware 2012) where CPU cores available can be shared among virtual machines.
iii. The design architecture in Zhikui *et al*(2009) does not mimic a typical datacenter.
iv. Approach failed to considerhidden request/transaction been process by the CPU especially on windows guest operating system where Antivirus and other services are running.
v. Zhikui et al (2009) model was not experimented online on the production server to accommodate real user experience.
vi. The model assumed that request is held only at one tier.
vii. Measurement of response time on the client is associated with too many factors that affects performance (network availability and speed, client system and its configuration, geographical location).

This research intends to solve these challenges mentioned above.

1.3 JUSTIFICATION FOR CASE STUDY

As application portfolios expands to meet up with the target of making virtually all processes IT based; budgets have tightened as well. The best approachis to increase server utilization and reduce costs using virtualization strategies that consolidate servers and pool IT resources allowing better control while increasing the flexibility of the IT infrastructure (Krishnamurthy *et al*., 2006).

One of the challenges of this technology lies in the fact that many of the application servers which in themselves are easily virtualized are attached to a myriad of storage devices that vary greatly in type and size. Monitoring the health of the virtual environment is very crucial as there is an expressed concern that the large increase in traffic to the data center may affect the overall performance experienced by the end user. Network bandwidth and overload virtual environment have also been cited as concerns that affects response by end users.

Datacenter managers therefore have the responsibility to monitor user experience and also monitor the technical environment to determine if performance in terms of application response time begins to deteriorate in order to respond to issues which negatively impacts the users' experience.This condition prompts questions like;how can we efficiently manage server resources despite highly varying applicationworkloads? This research attempts to provide solution to these issue.

1.4SIGNIFICANCE OF STUDY

The primary goal of the service level objectives (SLO) in virtualized data center is to ensure appropriate performance of the corresponding IT service.

Most previous researches have considered only an average performance (e.g., mean response time). However, average performance guarantees are not sufficient for many applications, especially the interactive ones. Hence, this research focuses on improving Zhikui et al (2009) model that optimizes performance of applications in virtualized data center. This research model is important for the following reasons:

(i) Capacity provisioning: Thiswill enable a server farm to determine how much CPU capacity to allocate to an application in order for it to service its peak workload.

(ii) Performance prediction: Enables the response time of the application to be determined for a given workload and a given hardware and software configuration

(iii) Application configuration:which enables various configuration parameters of the application to be determined for a certain performance goal

(iv) Bottleneck identification and tuning: which enables system bottlenecks to be identified for purposes of tuning

1.5RESEARCH QUESTIONS

i. How application-level performance in terms of response time and throughput are affected in virtualized system.

ii. How can we efficiently manage server CPU resource despite highly varying application workloads? (The data center environment makes these challenges particularly difficult since it requires solutions with high scalability and extreme speed to respond quickly to fluctuating internet workloads).

iii. What effect has resource management on application performance?(The CPU and memory allocated to a virtual machine can be dynamically adjusted and hence it is important to know the impact of this flexibility attribute on application performance).

iv. To what extent can CPU contention affect application performance? (Due to the consolidation nature of data centers, resource contention is intensified and there is need to minimize the resources used for the different applications while satisfying service level objectives).

1.6AIM AND OBJECTIVES

The aim of the research is to model the performance of virtual machines base on workload transaction/request history and allocated CPU in VMware Hypervisors.

The objectives of this research are to:

i. Deploy a typical datacenter to measure performance in a consolidated production environment using VMware hypervisor.
ii. To propose a model based on VMware hypervisors and Zhikui *et al*(2009) model
iii. To improvethe reliability of performance model by considering the number of cores available to virtual machines.
iv. To test the reliability of improved model using various testbed/platform and workload generators.
v. Identify the influence of CPU allocation on application performance in virtual machines in offline and live environment.

1.6METHODOLOGY

To model the performance of virtual machines, the researcher deploys a typical virtualized datacenter setup using VMware EXSI 5.1 Hypervisor running onHP SL170s, 1U 6 Intel Xeon 5600 server with 32GB RAM, 2.93GHz 2 4-core Intel Xeon Processor, 2x1TB HDD internal storage with 12TB SAN storage server of 12 RAIDs connected with NIC teaming configured to mimic an average datacenter solution.The CPU controlled approach is adopted such that CPU allocation to the virtual machine is varied as workload varies.All virtual machines were assigned the same number of CPU cores available on the physical host.

The methodology requires three different applications as workload generator setup on three testbeds with the same configuration. These workloads include a PHP and Java controller

based RUBiS standard workload generator for offline estimation of parameters and two custom application exMaster and AppSton based on ASP.net and PHP respectively for online parameter estimation on the production server. Performance data collectedfrom the application is used to validate the model.

Chapter Two

LITERATURE REVIEW

2.1INTRODUCTION

This chapter discusses the background of virtualization technologies as well as its roles in data centers. It goes on to uncover the weaknesses and challenges associated with virtualized data centers in relation to application performance. Researches that cover the factors which influence system performance and theories that optimize system performance are reviewed to set the context for the researcher's contributions.

2.2DATA CENTERS AND ITS CHALLENGES

There has been tremendous growth in the popularity of data centers as the processing power required by businesses exceeds the maintenance they can offer within their own corporate infrastructure. Data centers acts as large server farms that provide computation and storage resources to one or more businesses with greater efficiency using economies of scale. Data centers can be owned and operated by a business or a data center may be operated by a service provider which in turn rents shares of its resources to other businesses (Chandra *et al.*, 2003).

Operators of data centers face challenges right from the initial capacity planning stages of new applications deployment and provisioning, to efficient allocation of resources to meet the application performance guarantees of live systems. Maintenance of hardware reliability, cooling and energy needs of running several servers are also problems that must be dealt with. All of these issues are compounded by both the massive scale of modern data centers, and the fast paced workload dynamics of many data center applications (Raghanvendra *et al.*, 2008).

Table 2.1identifies key problem areas encountered by datacenter operators and some corresponding researchers that have worked on them. Administrators must first handle infrastructural challenges such as determining the architectures of the data center and provision of power and sufficient cooling for large number of servers.

Table 2.1 Summary of Previous Work in Virtualization

AREA	**CHALLENGES**	**RESEARCHERS**
Infrastructure	Server & Network Architecture, Cooling & Power Management	Church *et al*., 2008; Lim *et al*., 2008; Raghavendra *et al*., 2008; Chase *et al*., 2001.
Deployment	Capacity Planning, Service Placement, Application Modeling.	Zhang *et al*., 2007; Stewart *et al*., 2008; Ugaonkar *et al*., 2007.
Resource Management	Monitoring, Storage, Server, Network Provisioning	Chandra *et al*., 2003; Wester *et al*., 2009; Liu *et al*., 2007, Wang*et al*2009
Reliability	Security, Fault Tolerance, Availability	Pinheiro *et al*., 2007; Gupta *et al*., 2006; Garfinkel *et al*., 2003
Applications	Performance, Clustering Frameworks and Configuration Management.	Isard *et al*., 2007; Apte *et al*., 2007; Ugaonkar *et al*., 2007; Padala *et al*., 2007; Liu *et al*., 2007; Wang *et al*., 2009

There are problems associated with deployment and planning that relates to estimation of data center's capacity and initial provisioning for new applications (Zhang *et al*., 2007). This may require models of an application's resource requirements and an understanding of how they are impacted by different hardware configurations (Stewart *et al*., 2008). With data

centers attemptin improving resource utilization through the consolidation of servers, it has become necessary for operators of data centers to understand how applications placement impacts performance and resource consumption (Urgaonkar *et al*., 2007).

Efficiency in resource management is a key concern for data center operators whose aims are, meeting application Service Level Agreement (SLAs) and reduce costs. Multiplexing physical resources between multiple applications can be achieved via shared hosting platforms (Chandra *et al*., 2003). It is, however, worthy to note that provision of strong isolation between applications is very difficult without virtualization; and operating systems would have to be modified for fair allocation of resources (Wester *et al.*, 2009).

Reliability becomes a critical issue of concern when running vital applications within data centers. Due to the large scale of modern data centers, hardware components fail constantly, requiring high level reliability mechanisms within applications as well as low level fault tolerance techniques like RAID (Pinheiro *et al*., 2007).

The issue of security is crucial especially for shared data center environments, leading to much research on building trusted platforms and isolating services (Gupta *et al.*, 2006; Garfinkel *et al.*, 2003).

The necessity of clustering web servers and databases becomes obvious when a single commodity server fails to meet customer demands. Large scale data mining is also an increasingly popular use for data centers, with search engines becoming some of the largest consumers of data center resources. These systems employ clustering frameworks like Dryad and MapReduce to distribute work across many hundreds or thousands of nodes (Isard *et al*., 2007).

2.3SERVER VIRTUALIZATION

Virtualization became more eminent as a result of its aim in improving resource utilization through server consolidation. Virtualization can be performed at the application level e.g., VMware's Player software or Oracle's VirtualBox; within an operating system e.g., Linux Vservers or even below the operating system e.g., VMware ESX(VMware, 2012)

VMware's ESX platform is a full virtualization technique that provides a "bare-metal" hypervisor that manages a set of virtual machines running unmodified operating systems.
Xen uses a paravirtualization technique that requires small changes to be applied to the operating systems within each VM and a host OS to run device drivers, but allows for a simpler hypervisor layer. Both systems support fine grain management of memory and CPU resources, as well as the ability to transparently migrate running virtual machines from one physical server to another(Nelson *et al*., 2005).

2.4VIRTUALIZATION IN DATA CENTERS

Server virtualization has gained popularity in data centers since it provides an easy mechanism to partition physical resources cleanly, allowing multiple applications run in isolation on a single server. Virtualization also helps with server consolidation and provides flexible resource management mechanisms, but it can introduce new challenges. The ability to determine where to run applications in a shared environment remains a challenge (Ugaonkar *et al*., 2007), and virtualization adds more difficulties as a result of the variable virtualization overheads seen by different applications on different platforms. Some commercial systems now exist for automating the management of VM resources and a variety of research projects have proposed schemes for management of processing (Grit *et al*., 2006; Ruth *et al*., 2006) and memory resources (Zhao et al., 2009; Lu *et al*., 2007).

2.5UNDERSTANDING VMWARE HYPERVISORS

Every hypervisor is different by design. A model of one system with a given hypervisor may not accurately model another system with a different hypervisor. Many research groups use the Xen hypervisor because it is open source and supports both full virtualization and para-virtualization. (VMmark 2012)

VMware proprietary ESXi hypervisor, which serves as a cornerstone for their vSphere cloud computing platform, provides a host of capabilities not currently available with any other hypervisors. These capabilities include high availability (the ability to recover virtual machines quickly in the event of a physical server failure), Distributed Resource Scheduling (automated load balancing across a cluster of ESXi servers), Distributed Power Management (automated decommissioning of unneeded servers during non-peak periods), Fault Tolerance (zero-downtime services even in the event of hardware failure), and Site Recovery Manager (the ability to automatically recover virtual environments in a different physical location should an entire datacenter outage occur). (VMmark 2012).

2.6EVALUATING SYSTEM PERFORMANCE IN DATACENTERS

Apte *et al*. (2007) carried out a research on how CPU allocation and system response time can be balanced in a virtualized data center. In this research, a control-theoretic approach was used to tune the CPU resource allocated to a virtual machine in order to optimize the application performance metrics while using minimal CPU resource. The approach used does not expect any input for performance targets to steer the system towards optimal values of CPU share allocations and system performance. The prototype and validation of the methodology was done on a small virtualized testbed via the use of Xen virtualization environment.

The control-theoretic resource tuning approach was used on a simple server consolidation scheme which triggered VM migrations and powers servers up and down. The preliminary results showed that the controller allocated CPU share optimally and worked well with a server consolidation scheme to achieve good application performance.

A number of researchers have addressed the problem of determining resource allocation of virtual machines in recent past. Among the early contributions, is the approach by Padala *et al* (2007) which proposed a two-layered controller architecture. This approach maintains the CPU share utilization of a VM at a given level, and meets a relative QoS metric between VMs.

Liu *et al* (2007)also presented a control theoretic approach which included an onlinesystem model estimation mechanism (for estimating model parameters A and Bunder changing conditions), and an LQR-based algorithm to derive controller gain K. The controllers' approach used seems workable; however, it would have been more meaningful to estimate model parameters offline and do not assume performance targets for resource allocation decisions. Also, it is important to understand the relationship between virtualized CPU resources, CPU contention and the response time which would be one basic contribution of this research so that system performance can be reasonably predicted for efficiency to be attained by autonomic controllers to satisfy service level objectives.

Wang *et al*(2009) presented a coordinated power and performance control mechanism which again uses two control loops. The "outer" control loop maintains a specified power budget by manipulating the CPU frequency levels, while an "inner" control loop adjusts the CPU resource allocated to a VM to maintain application performance at a high level. A

reinforcement learning based approach has been proposed by Rao *et al* (2009) to determine the CPU and memory slices to be allocated to a VM. The "actions" in the model are step changes in these allocations, and the reward is related to high application throughput and low application response times.

Zhikui *et al* (2009) presented a performance modelbased on workload transaction history using Zen hypervisor and RUBiS. The model can predict performance of multi-tier applicationsrunning on virtualized servers with variable CPU entitlements. The modelling approach is non-intrusive in the sense that the process of modelparameter identification requires no additional instrumentation and the data used in this approach is readily available fromstandard system and application monitoring. Furthermore, the modelcaptures the resource demand at per-transaction type and per resourcelevels and can be used to derive resource demandfor different transaction mixes. Hence this approach can beapplied to realistic workloads where both the frequencies oftransaction types and request volume rates can change overtime. However, the model has many drawbacks such as workload estimation on offline server that did not capture real user experience, experimental design approach does not mimic a typical datacenter, Hypervisors like that allows CPU cores to be assigned to VMswere not considered,server utilization was not consolidated, model parameters does not capture hidden workloads that consumes CPU time like antivirus and other services on windows guest operating system, the model assumes that resources areheld at exactly one tier at a time as a request proceeds throughthe system.

In handling the problem of dynamisms of workloads in real applications, Chen *et al* (2009), proposed a practical approach that combined fine grained performance models with regression analysis to translate service level objectives into design and operational policies for multi-tier

applications. Even though the approach used seem to cope with complexity in applications, the research was carried out on physical servers. To cope with the problem of diversity in applications in virtual data centers will require a model that will generally capture the applications' behavior with different configurations, CPU allocation and system performance.

2.7THEORIES ON SYSTEM PERFORMANCE

This research aims at developing an accurate and intuitive method for modeling systems' performance in order for system administrators and managers to have confidence in the control of systems. Although theories themselves are not the solutions in explicit terms but they give directions that point at variables which are very important in solution finding. In order to derive the framework for this research, three established theories were considered.

2.7.1 QUEUING THEORY

A number of researchers have used queuing theory to model the behavior of computer systems. Some recent research efforts used the queuing performance models for interactive and enterprise applications such as multi-tier internet applications (Zhikui *et al*, 2009; Chen *et al*, 2009; Zhang *et al*, 2007; Stewart *et al*, 2007; Ugaonkar *et al*, 2005). The model proposed by Urgaonkar *et al* (2005) is well known. It uses a closed queuing network model and the Mean Value Analysis algorithm for multi-tier applications. The models though seem reliable in some scenarios, most of the queuing models however require detailed service demand information that is not readily available from traditional monitoring. Also, the parameters the model uses were normally derived for specific workloads; hence the research results are reliable only for similar systems with such transaction mix.

The queuing model by Stewart *et al* (2007) employed an open queuing network model that explicitly models a non-stationary transaction mix. Deriving the model parameters can be

done from monitoring data through regression analysis, but sufficient application knowledge is required to measure transaction mix. Chen *et al* (2009) generalized the ideal and developed models for both session-based and request-based workloads. High resource utilization seems to be ignored by the approach of most queuing models. Although Urgaonkar *et al* (2005) considered congestion effects in their model but the general performance seems not to be clear. Most prior work keyed their focus on physical server platforms rather than virtualized environments and of course the models generated for such physical servers may not suitably fit the virtualized environment.

2.7.2 USE OF PERCENTILES

The use of percentiles in modern IT services has gained popularity over the traditional performance model of the use of moments e.g., mean response time. Price (1976) presented percentiles as a preference over moments due to the weakness observed in using moments in modeling computer systems. Amazon used the 99.9^{th} percentile in imposing stringent requirements on the latency of their platform (DeCandia *et al*, 2007). Other researchers that adopted the percentiles instead of moments include; Lublin and Feitelson (2003), and Bodik *et al* (2007) among others. Rather than working with the percentiles, some other researchers dived into opportunities to improve accuracy in queuing models.

2.7.3 STATISTICAL BASED MODELS

The application of statistical learning and probability modeling to performance management is on the increase. Cohen *et al* (2005) presented an approach that correlates SLO violations with system signatures. Kumar *et al* (2008) presented an approach that partitions the system's state-space into homogeneous subspaces, for each subspace a micro-model was created and the micro-models were used to translate higher-level objectives into component-level

objectives. However, none of these statistical approaches focuses on general performance modeling.

2.8RESEARCH MODEL

Virtualization layers increase the challenges of performance modeling, because different applications may compete with one another for resources in complicated ways across many different servers. Based on the study and requirement of queuing theories and model by Stewart *et al* (2009), an improvement to Zhikui's model has been proposed.

Compared to previous solutions, this methodology evaluates and models the tail distributions of performance and works well for different scenarios with low and high CPU utilizations in virtualized environments taking into consideration unknown applications running underground, multi-threading in multi-core processors as well as different OS platform and application design technology. Also, a typical datacenter design has been considered and the proposed model is calibrated from data that is generally available from regular system resource and application performance monitoring. Though this research approach can incorporate multiple resources; memory, network, and disk I/O will not be included in the evaluation.

Table 2.2 Summary of Related Work in Performance Modelling of Virtual Machines

Work	**Analytical approach**	**Application type**	**Server concurrency**	**Level of Granularity**	**Server Type**
Urgaonkar et al (2009)	Closed network of queues	Multi-tier	Single threaded	Request	Hardware
Apte *et al (2009)*	Control-theoretic, Linear Quadratic Regulator	Multi-tier	Single threaded	Transaction Mix	VM
Zhang et al. (2007)	Closed network of queues	Multi-tier	Single threaded	Transaction	VM
Padala *et al*	two-layered controller architecture	Multi-tier	Single threaded	Transaction Mix	VM
X. Wang (2009)	Control Loop	Multi-tier	Single threaded	Transaction	VM
Rao *et al (2009)*	Performance model based onstep functions	Multi-tier	Multi-threaded	Request	VM
Zhang and Fan	Queuing model	Single tier	Single threaded	Request	Hardware
Chen et al. (2009)	Closed multi-station queuing Network	Multi-tier	Multi-threaded	Request	VM
Zhikui et al	Queuing model, Control Theory	Multi-tier	Single threaded	Transaction	VM
Present Work	Queuing model, Control Theory	Three-tier	Multi-threaded	Request	VM Consolidated

Chapter Three

MATERIALS AND METHODS

3.1INTRODUCTION

In an effort to improve virtualized systems, researchers look to modeling and simulation techniques in order to pinpoint performance bottlenecks and resource contention. Many challenges unfold in the development of accurate performance models, as virtualized environments add levels of complexity beyond models suitable for single physical machines

By design, virtual machines share resources and thus compete for resources. It is difficult to accurately model a system with resource contention when many factors, including number of virtual machines, type of hypervisor, differences in underlying and support hardware, type of application, and overall workload characteristics can vary widely even in what appear to be otherwise homogenous environments. These components and the numerous interactions between them can affect performance in ways that are not yet fully understood. Specific challenges with performance analysis of virtualized environments are hypervisor differences, resource interdependence, workload variability, and disparity in modeling.

3.2ASSUMPTIONS

There are fundamental measurement issues when guest machine requests to read the hardware clock and set hardware-based timer interrupts that are virtualized. Since running as a virtual machine is transparent to the guest Operating System (OS) – with some exceptions, measurements taken from inside the virtual machine guest are unaware of any delays introduced due to resource sharing. The accuracy of internal measurements of application response time is affected by the virtualization environment, which virtualizes all the clock and timer services that the guest machine uses.

Consequently, measurements taken from the guest machine cannot reliably quantify queuing delays associated with queuing for virtual processors. These delays may be negligible depending on the hypervisor design. The contention problem is reduced in Vmware; this is handled by the dynamic resource scheduler (DRS) to load balance shared resources.

A general methodology for constructing performance models that require little special knowledge of the application is considered. This model is then demonstrated to ascertain the validity of the methodology through experiments in virtualized environments across a range of resource allocation and contention states to show that this methodology can model the probability distribution of response time with a mean absolute error of less than 5% when compared with the measured response time distribution. An application of these models to case studies that investigate the impact of CPU allocation and contention on application performance can then be examined.

In this work, a performance model is developed, in which the researcher models the response time distribution of an application performance metric conditioned on variables that can be measured or controlled, such as system resource utilization and allocation metrics. The influence of hidden factors that cannot readily be measured are considered to be implicit in the distribution of performance.

Thus, non-CPU resources are assumed to be adequately provisioned and hence the effect of contention for these resources on the response time (i. e., the queuing delay) is negligible.

These issues were considered hidden factors and the following assumptions were also made.

i. CPU is the only resource to be dynamically allocated among virtual machines.
ii. Virtual Machine scheduler employs a capped mode, such that a virtual machine cannot use more than the CPU time allocated to it. This assures a straightforward guarantee on resource allocation and provides good performance differentiation between applications sharing physical resources.
iii. The virtual machines will always have the same CPU allocation.
iv. Applications are monitored by their response time and CPU consumption.
v. Poisson process is a good approximation of request arrivals
vi. All virtual machine data are stored in a LUNs on the SAN (storage server).

3.3EXPERIMENTAL SETUP

This experiment consists of two experiments based on applications used as workload generators. The two testbeds consist of virtualized environment configured with Vmware Exsi 5.1 Hypervisor running onHP SL170s, 1U 6 Intel Xeon 5600 server with 32GB RAM, 2.93GHz 2 4-core Intel Xeon Processor, 2x1TB HDD internal storage with 12TB SAN storage server of 12 RAIDs connected with NIC teaming configured to mimic an average datacenter solution. All virtual machines were assigned the same number of CPU cores available on the physical host.

3.3.1 EXPERMENTAL SETUP USING STANDARD RUBIS WORKLOAD

This section considered a standard workload generator basedon PHP RUBiS (Rice University Billing System) Benchmark as the workload generator for offline parameter estimation.

RUBiSis an open source benchmark application designed for several platforms (PHP, JAVA, etc) with MySQL as the database server and java based client system as the controller.

In this experiment, each web tier, application tier and database tier runs in an individual virtual machine A,B, and Crunning a Linux (2012 Obuntu server) as guest operating system on the same physical server as shown in fig 3.0

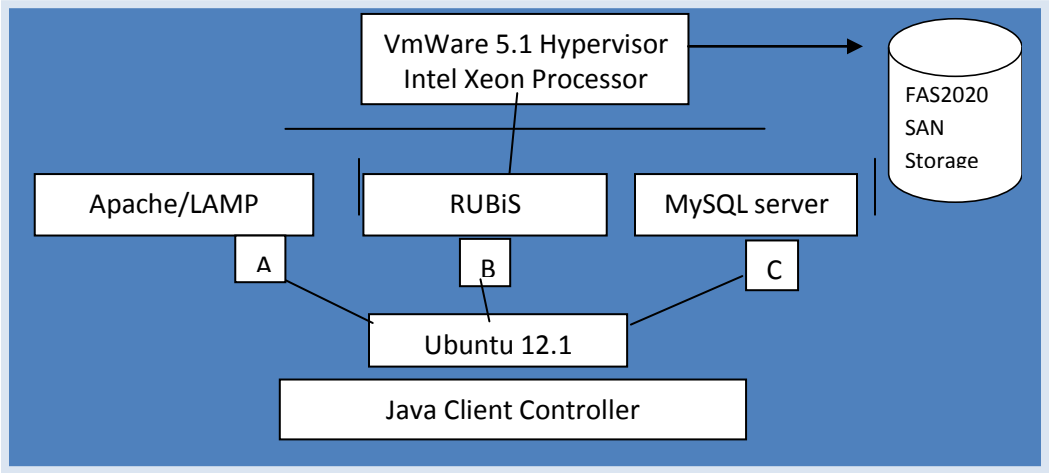


Fig 3.0: Experiment Setup in Virtualized Environment for Standard Workload

Five (5) points of variation for CPU allocation to the two virtual machines as were considered as follows: 20%, 40%, 60%, 80% and 100%. At intervals of 60 seconds, the performance metrics was recorded. 50 measurements were taken for the different CPU allocations for the two virtual machines (web and database server) while varying the number of threads (100,200,300, 400 and 500) running on the client generating workload.

3.3.2 EXPERMENTAL SETUP USING CUSTOM WORKLOAD

In this section, the researcher deployed custom applications as workload generators. The custom workload generators are based on two applications with different technology platform to estimate parameters on the production server. This is to enable the researcher monitor real

user experience as CPU allocations are varied at 20%, 40%, 60%, 80% and 100%. The applications are:

- exMaster based on ASP.Net is designed using C# and MySQL database server. exMaster is an online result processing system designed for the University of Agriculture Makurdi to manage student academics records.
- AppSton based on Apache is designed using PHP and MySQL database server. AppSton is a comprehensive student portal designed for University of Agriculture Makurdi to manage student records from admission, payment of fees and registration of courses.

At peak periods when the various users(students, registration officers, examination officers and banks) of the applications are busy (usually during registration processes), CPU allocations are varied and performance metrics are recorded at an intervals of 60sec. To collect data, MySQL monitoring agent was employed to periodically collect CPU consumption values for each virtual machine in which an application component is running.

The webserver monitoring tool in IIS was employed to monitor response time for the transactions. IIS was configured to log HTTP site activity by using the IIS log file format. A sample IIS log description on the web server is given in Table 3.0

Table 3.0: IIS Log Description on the Web Server

Fields	Value	Descriptions
Client IP address	192.168.114.201	The IP address of the client.
User name	-	The user is anonymous.
Date	03/20/12	This log file entry was made on March 20, 2001.

Time	7:55:20	This log file entry was recorded at 7:55 A.M.
Service and instance	W3SVC2	This is a Web site, and the site instance is 2.
Server name	Banya	The name of the server.
Server IP	172.21.13.45	The IP address of the server.
Time taken	4502	This action took 4,502 milliseconds.
Client bytes sent	163	The number of bytes sent from the client to the server.
Server bytes sent	3223	The number of bytes sent from the server to the client.
Service status code	200	The request was fulfilled successfully.
Windows status code	0	The request was fulfilled successfully.
Request type	GET	The user issued a **GET**, or download, command.
Target of operation	/DeptLogo.gif	The user wanted to download the DeptLogo.gif file.
Parameters	-	There were no parameters passed.

The time-taken field measures the length of time in miliseconds that it takes for a request to be processed. The client-request time stamp is initialized when HTTP.sys receives the first byte of the request. HTTP.sys is the kernel-mode component that is responsible for HTTP logging for IIS activity. The client-request time stamp is initialized before HTTP.sys begins parsing the request and client-request time stamp is stopped when the last IIS response send completion occurs.

Workload estimation is based on the number of requests available to the webserver when metrics are recorded. Although this approach records high level of workload fluctuation based on user activity; this is to capture real user experience in a production environment. The IIS

worker process logs and Apache server logs records the values for the ASP.Net and PHP based applications respectively.

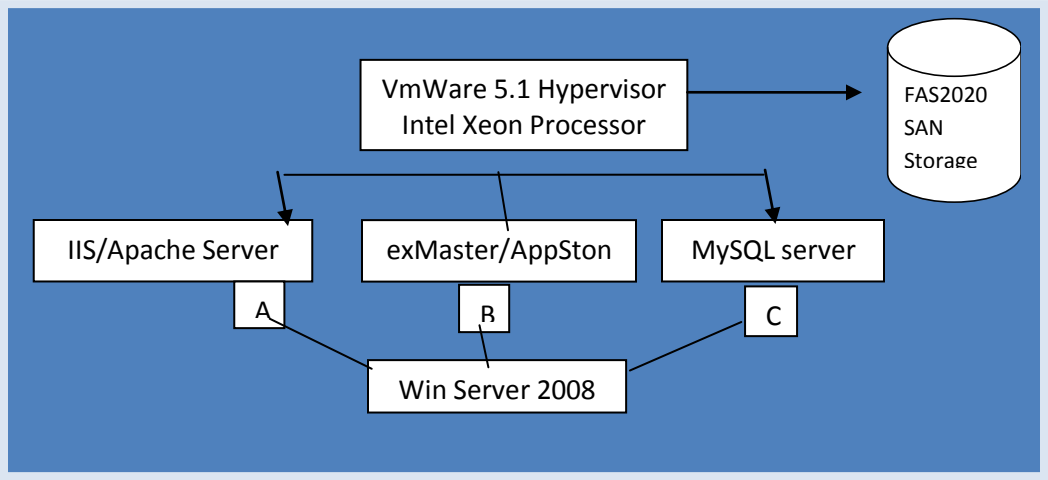


Fig 3.1: Experiment Setup in Virtualized Environment for Custom Applications

Given a specific application and transaction/workload, the performance is affected by the resources allocated to the virtual machines hosting the application and the type of application, as well as less readily observable factors like system scheduling, cache misses considering the fact that an external storage is been employed, and interactions between application components, Speed of network accessing the storage device as well as disk latency, Queues that arises from software resources, such as threads, database locks, and protocol ports.

Although these hidden factors are not considered explicitly, they do have an effect on the distribution of a performance metric like response time, and some of them may be partially correlated with the variables that are considered in this model.

Here are the notations and basic definitions used:

i. Resource consumption c is the actual percentage of the physical resource consumed by a virtual machine during a given time interval.

ii. α Resource allocation to refer to the percentage of a physical resource capacity

iii. U Resource utilization

iv. Ω Number of CPU cores available to virtual machine

v. n, N Number of virtual machines running separate tiers.

vi. i , Itransactions types

vii. T mean Response time for CPU resource

viii. β is the request from known applications running on the server

ix. δ is the request from unknown applications running on the server

x. λ_{in} average CPU demand of transaction i in tier n

xi. γ is the workload

3.4 DATA COLLECTION

To collect data, utility tools (iis, logs, apache logs and Qperforma) on the servers was used to extract utilization and response time details. MySQL monitoring agent was employed to periodically collect CPU consumption and average response valuesfrom virtual machine hosting the database server while APACHE server logs and IIS worker process logs shows the utilization details on the webserver.

3.5 MODEL GENERATION

The approach employed here is to characterize response time as a probability distribution conditioned on allocation while focusing on the dependency of CPU allocation and response time as well as the contention rate of the allocated resources.

Recall that each tier (web, database) in the case studyof the experiment runs on separate virtual machine. This can be assumed as a general setup for a typical consolidated datacenter solution. Hence we have virtual machines V_n (n=1,2, 3...N).

According to Zhikui *et al (2009)*, the intensity of the workload can be defined as a vector ($\beta_1, \ldots, \beta_i$), where $\beta_i$is the average request rate of transaction type *i* during one time period for known applications.

Hence the aggregate rate of the known transaction/request type ican be defined as

$$\beta_n = \sum_{i=1}^{i} \beta_{ni} \qquad (3.1)$$

Similarly, following Zhikui's expression as stated above, workload for unknown can be defined as a vector ($\delta_1, \ldots, \delta_i$), where $\delta_i$is the average request rate of transaction type *i* during one time period for unknown applications. This unknown application request could be from underground services that has impact on the residence time.Since poison is a good approximation of request arrival rate, δcan be estimated using poison.

In the same vein the aggregate rate of the unknown transaction can be defined as

$$\delta_n = \sum_{i=1}^{i} \delta_{ni} \qquad (3.2)$$

Let the total workload from known and unknown transactions on tier n be defined as γ_n

$$\gamma_n = \beta_n + \delta_n \qquad (3.3)$$

Our earlier assumption is that non-CPU resource are adequately provisioned and hence the effect of contention for these resources on the response time (i. e., the queuing delay) is negligible.

According to Zhikui *et al*(2009)utilization is defined as the ratio of resource consumption and resource entitlement as

$$u = \frac{c}{\alpha} \qquad (3.4)$$

From queuing theory according to Virtamo (2003),the total CPU resident time by all the requests served in tier *n* is representedby$\frac{U_n}{1-\mathrm{U_n}}$ where $u_n$is the CPU utilization of tier *n*.

According to Ricardo (2011), Transaction type i may proceed to core j for processing with probability Ω (i, j) or leave the system. Assume single CPU core was allocated to the virtual machine, all server request would be handled by the same physical core. The routing probability from transaction of type i in VM n to the aggregated core subsystem can be represented as $0.9 < \sum_j \Omega(\mathrm{i,j}) < 1$ to model the higher CPU utilization that is observed when running virtualized applications.

Given that T is the reference service time (when using a single physical core), thus it can be established that the use of Ω (n) cores by VM n produces:$\frac{\gamma}{\Omega}$ as workload shared among cores in tier n

Since it is assumed that all virtual machines have the same number of CPU core and allocations, then the CPU resident time of all the transactions in tier n with aggregate β and δ is

$$\mathrm{T_n} = \frac{\gamma_n}{\Omega} \sum \frac{\mathrm{U}_n}{1 - \mathrm{U}_n} \qquad (3.5)$$

Cape mode is employed in our assumption; hence we can replace utilization u for response time Ƭ. From equations (3.4) and (3.5) thus,

$$\mathrm{T_n} = \frac{\gamma_n}{\Omega} \sum \frac{\mathrm{C}_n}{\alpha_\mathrm{n} - \mathrm{C}_n} \qquad (3.6)$$

According to Zhikui *et al* (2009), the resource demands of different transaction types are usually different, but the resource demand of a single transaction type is relatively fixed irrespective of the transaction mix of the workload and CPU entitlement of the virtual machines, since each transaction type usually has a relatively fixed code execution path and hence a stable resource demand.

Hence, the number of request that is generated at each tier is relative to the request rate of transaction issued by the user such that given a workload with transaction mix $\beta_1, \ldots, \beta_i$, the CPU consumption can be estimated as a linear function of the transaction mix.

$$C_n = \sum_{i=1}^{I} \lambda_{\mathrm{in}} (\beta_\mathrm{i} + \delta_\mathrm{i}) \qquad (3.7)$$

Thus equation (3.6) is the response time for CPU resource. Equation (3.7) is the utilization model.

In order to finalize the performance model, the target is to determine response time with respect to CPU allocation. Since response time for non-CPU service time is not affected by the allocated CPU or entitlement, we conclude that equation (3.6) is the performance model.

Chapter Four

RESULT AND DICUSSIONS

4.1 INTRODUCTION

The experimental setup and methodology of data collection as well as the proposed model were discussed in the previous chapter. This chapter will focus on the validation of the proposed model.

Virtual Machine A is configured to host theApache/IIS application on web server that runs on Apache/IIS with PHP/ASP.Netthe web server. Virtual Machine C is the database server with MySQL. The host VMware Hypervisor is connected to a 12TB FAS2020 SAN storage server.

To evaluate the model, data collected from the performance statistics of CPU allocation and consumptions as well as the number of transactions β in the application been considered are used (Appendix A, B and C). The focus is to compare the estimated values and the measured values of the response time for the different CPU allocations.

4.2 MODEL EVALUATION BASED ON STANDARD RUBiS WORKLOAD

The evaluation will consider several transactions mix that generate multiple requests/threads as allowed by the client workload generator. See "Apendix A" data on the performance metrics.

Evaluation process begins with our utilization model equation (3.7) in the previous chapter by using linear regression to fit the linear relationship between CPU consumption and workload (number of threads). Recall that a linear regression equation is established by the equation $y=a+b$xwhere y and x are variables, b is the slope and α is the regression constant.

Using sample data of the CPU allocation at 100% in the virtualized environment running PHP on Apache. The relationship was established using SPSS data analysis tool as shown in Table 4.1

Table 4.1 Sample Linear Regression Analysis for RUBiS Workload History

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	348.947	206.193		1.692	.097
	CPU	-3.947	16.547	-.034	-.239	.812

a. Dependent Variable: VAR00001

Hence from table above, it can be established that $C_n = 348.947 - 3.947\lambda$ however, the significance difference in the estimation of the parameters is greater than 0.05 which questions the linearity of the relationship as proposed by Zhikui *et al(2009)*.

To evaluate the performance model, the values of the observed response time and the estimated response time were compared as shown in the Figures 4.1 to 4.5

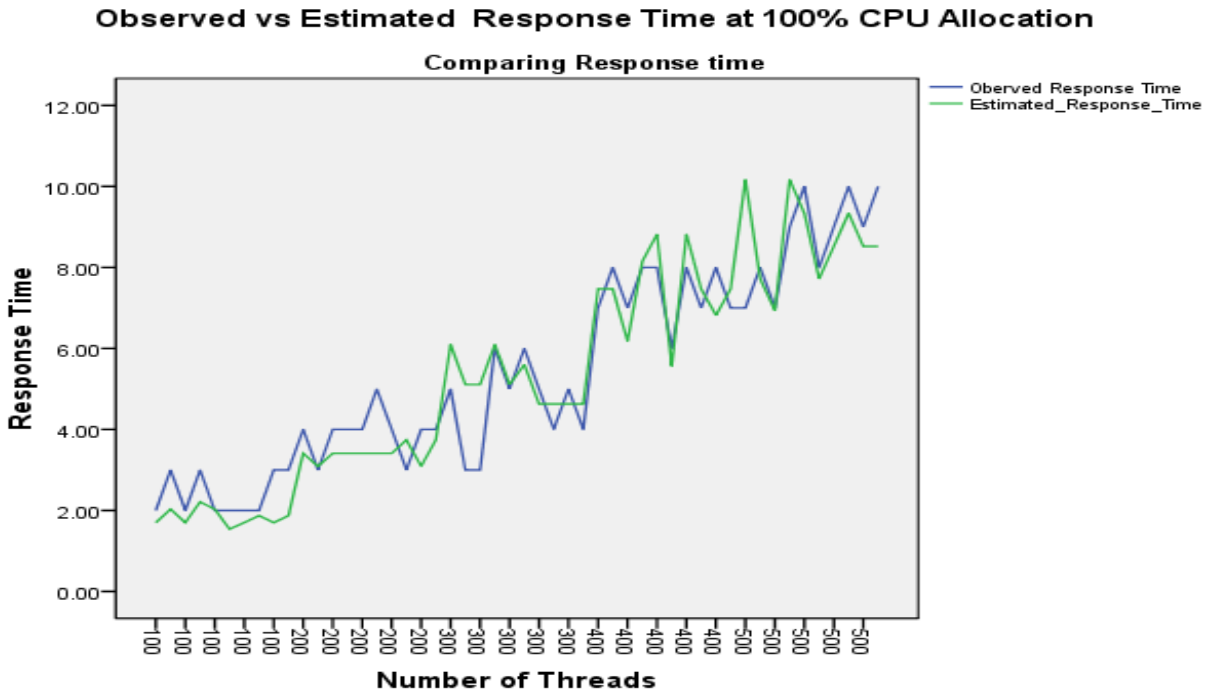


Fig 4.1: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 100%

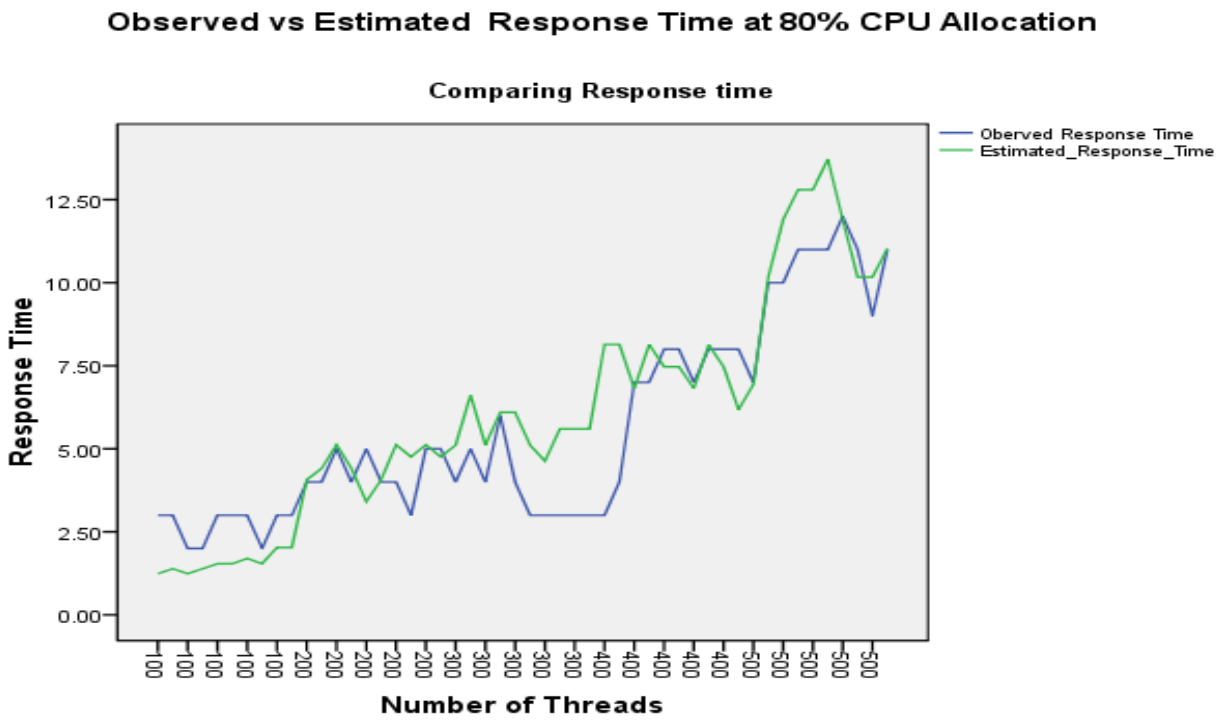


Fig 4.2: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 80%

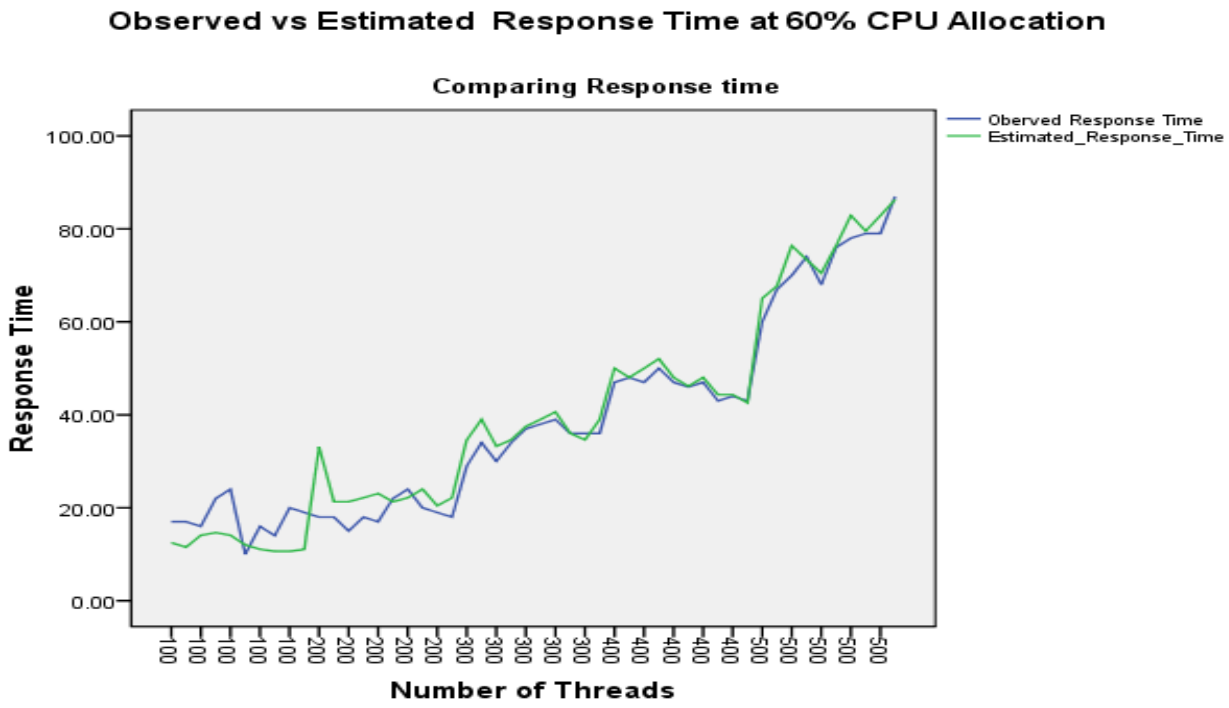


Fig 4.3: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 60%

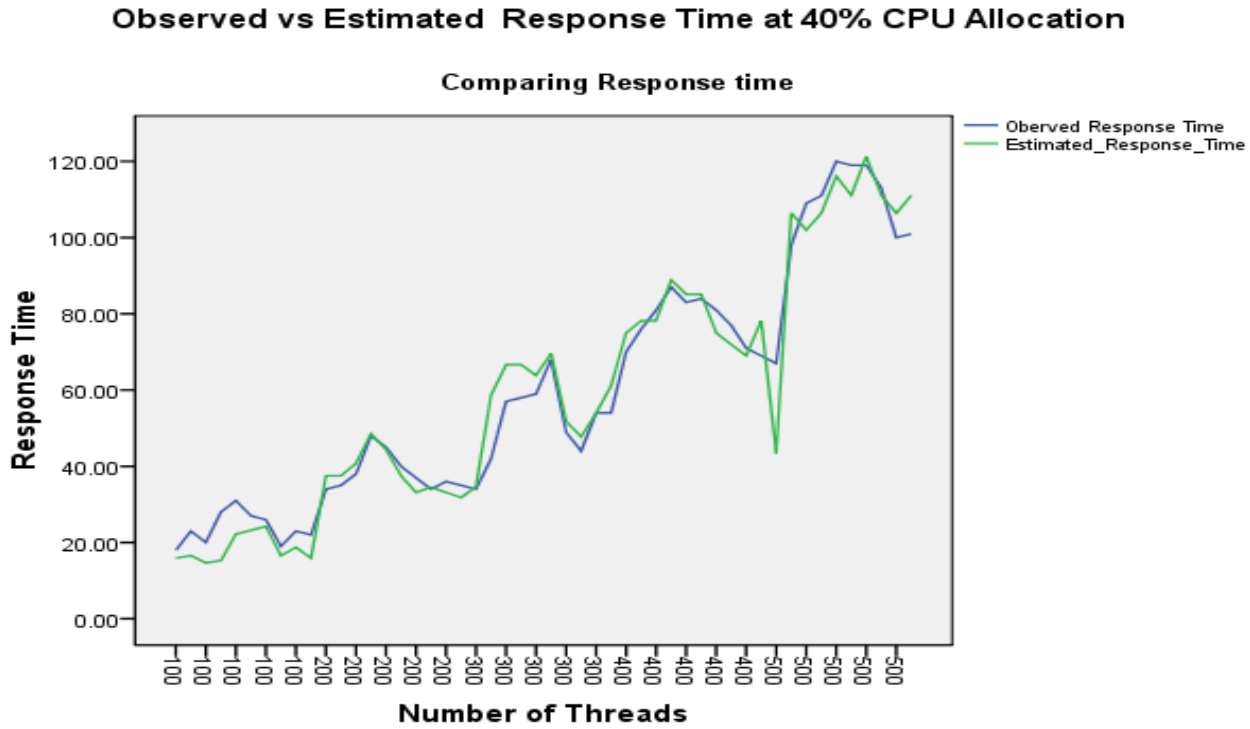


Fig 4.4: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 40%

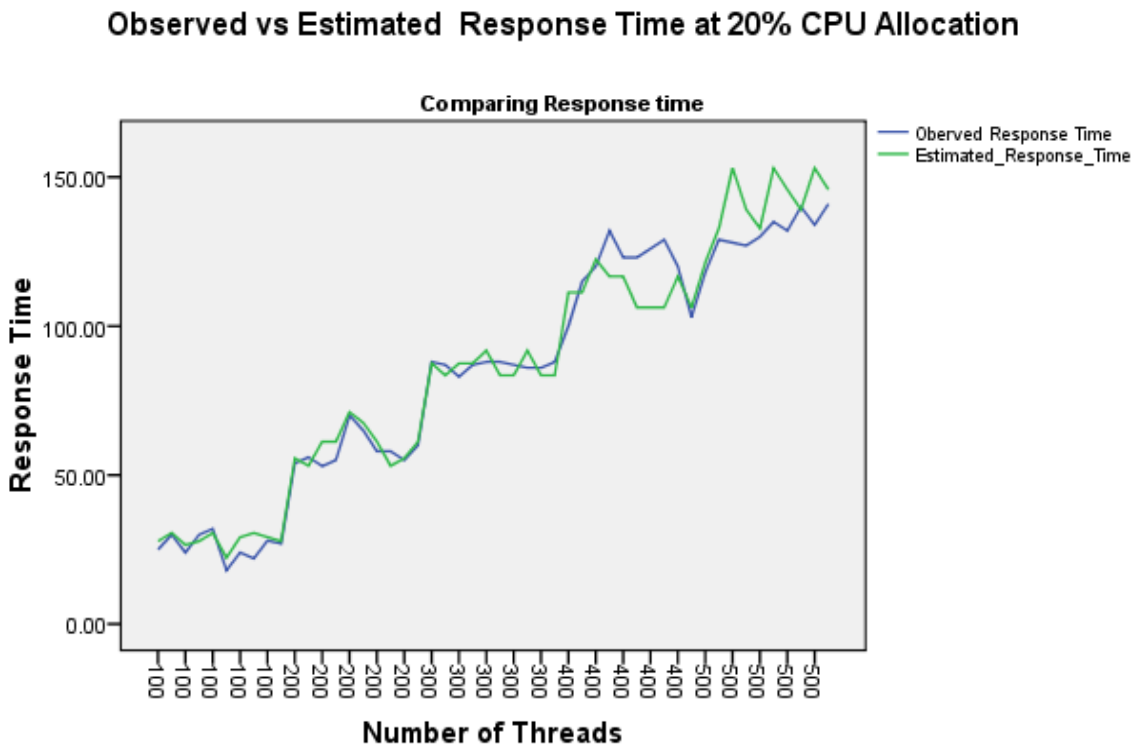


Fig 4.5: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 20%

From the figures above (Fig 4.1 - 4.5),it is observed that our performance model estimated value fits the observed/measured value as CPU allocation increases. The values are closely related at CPU allocations of 20 %, 40%, 60%, 80% and 100%.

To ascertain the validity of the model, we consider the mean response time % error in the model as shown in the Table 4.2

Table 4.2 MRT Estimation Error for RUBiS Application based on PHP and Java controller

% CPU Allocation	Avg Observed(s)	Avg Estimated (s)	% Error
20	84.34	85.66	1.565094
40	59.48	59.14	-0.57162
60	38.26	39.12	2.247778
80	5.84	6.06	3.767123
100	5.4	5.31	-1.66667

From the table 4.2, the estimated response time for allocations at 20%, 40%, 60%, 80%, and 100% are considerable close to the observed values for corresponding allocations with higher values recorded at 80% allocation since it is assumed that the workload for the unknown applications is zero. Workload from unknown applications cannot be readily observed due to background processes. It therefore infers that the model fits at the various CPU allocations of 20%, 40%, 60%, 80%, and 100%.

4.2 MODEL EVALUATION BASED ON ASP.Net and IIS

Using sample data of the CPU allocation at 100% (Appendix B) in the virtualized environment running exMaster application based on ASP.Net C# on IIS 7. The relationship was established using SPSS data analysis tool as shown in Table 4.2

Table 4.2 Sample Linear Regression Analysis in exMaster Performance Metrix (ASP.Net/IIS 7.0virtual machine)

Model		Unstandardized Coefficients		Standardized Coefficients	T	Sig.
		B	Std. Error	Beta		
1	(Constant)	54.558	2.467		22.111	.000
	CPU	-.134	.136	-.140	-.982	.331

Hence from table above, it can be established that $C_n = 54.5 - 0.134\lambda$. Similarly, as noted in table 4.1, the significant difference in the estimation of CPU parameter is greater than 0.05.

To evaluate the performance model, comparison of values of the observed response time and the estimated response time as shown in the Figures 4.6 to Figure 10.

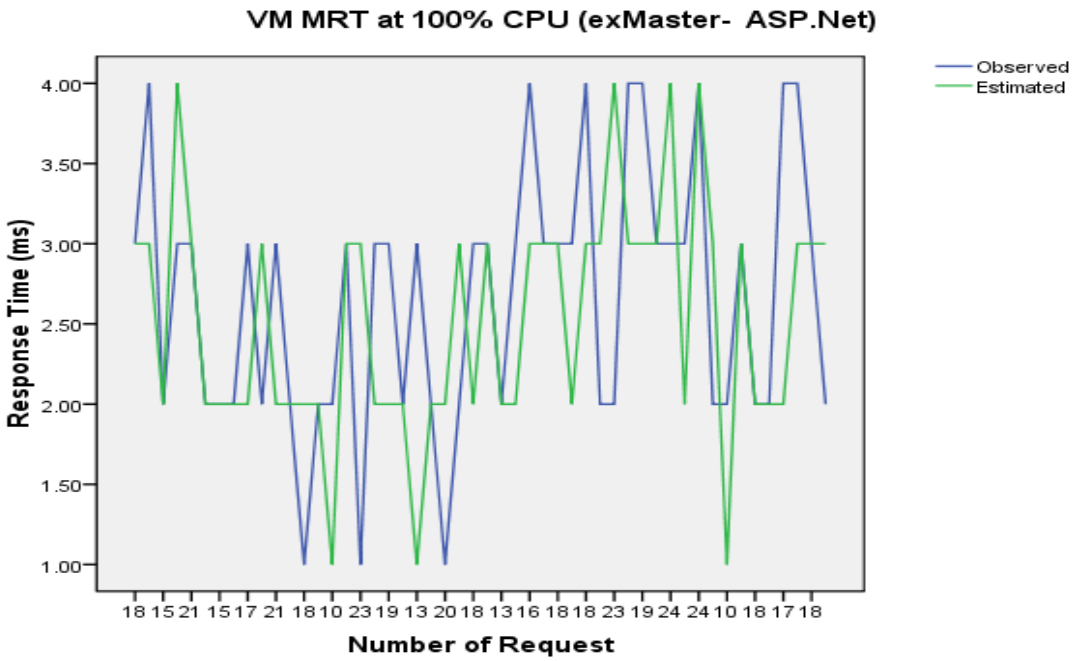


Fig 4.6: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 100% (ASP.Net, IIS & MySQL)

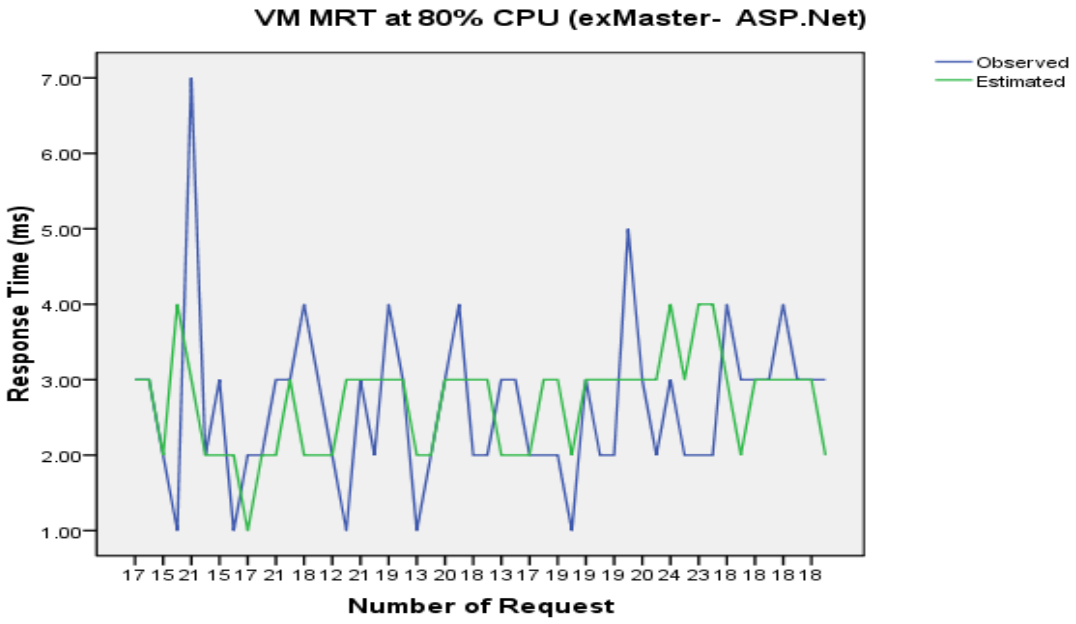


Fig 4.7: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 80% (ASP.Net, IIS & MySQL)

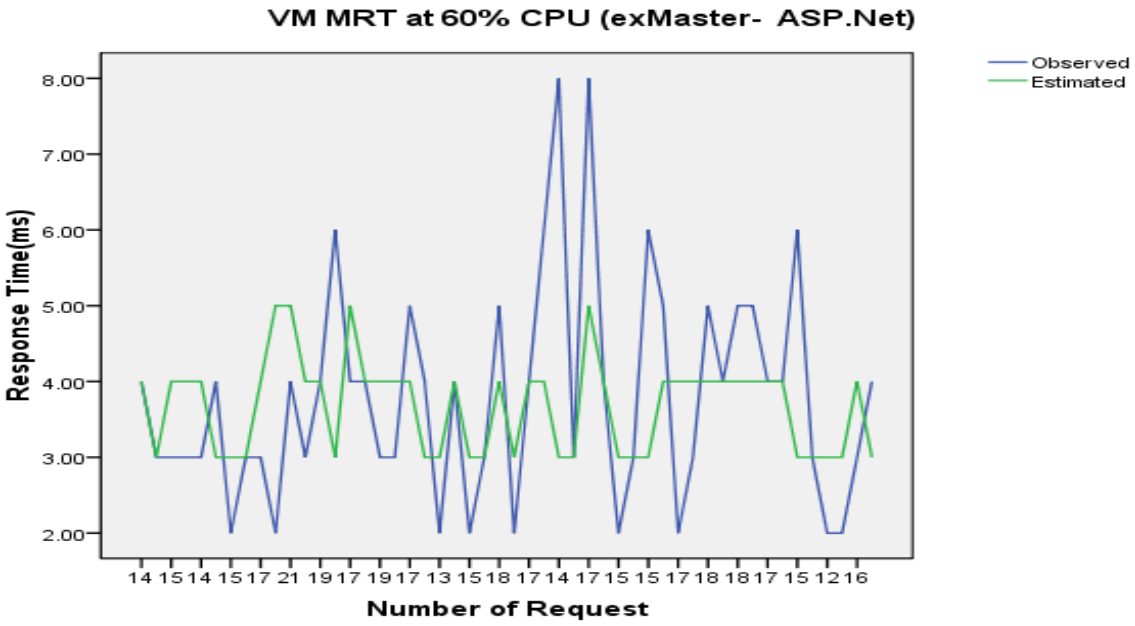


Fig 4.8: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 60% (ASP.Net, IIS & MySQL)

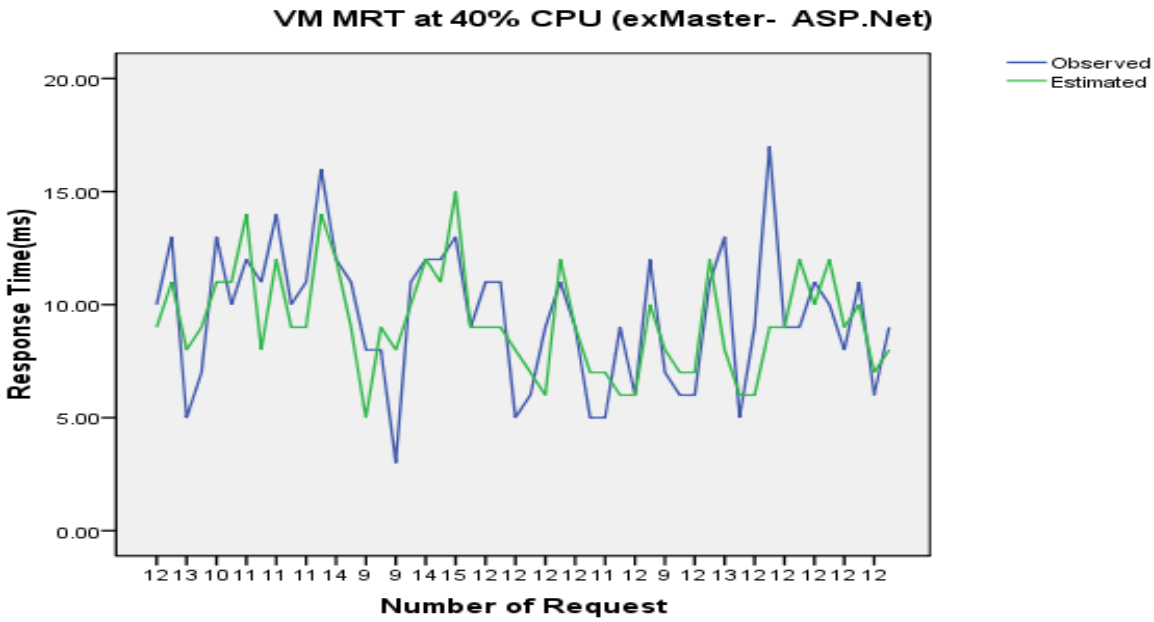


Fig 4.9: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 40% (ASP.Net, IIS, MySQL)

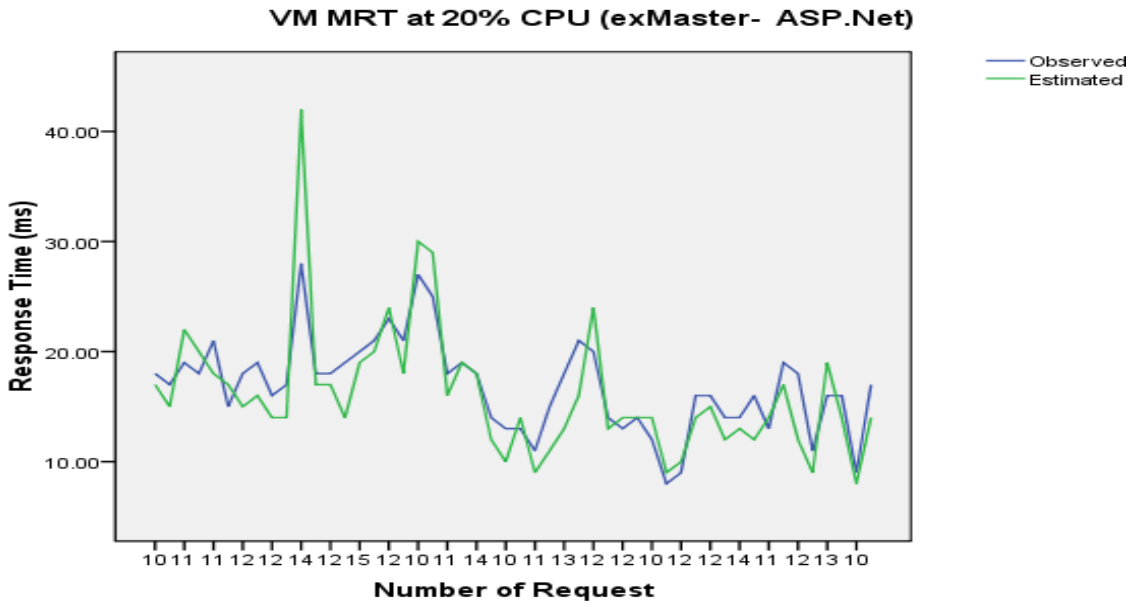


Fig 4.10: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 20% (ASP.Net, IIS, MySQL)

From the figures above (fig 4.6 - 4.10), in ASP.Net, IIS and MySQL platform test, it is observed that the performance model estimated value fits the observed/measured value at all CPU allocation.At CPU allocations of 60%, 80% and 100% the consumption tends to stabilize with lowering response time at varying request.

To ascertain the validity of our model, we consider the mean response time % error in our model as shown in the Table 4.4

Table 4.4 MRT Estimation Error for exMaster Application on ASP.Net

% CPU Allocation	Avg Observed(s)	Avg Estimated (s)	%Error
20	16.86	16.08	-4.6
40	9.54	9.19	-3.6
60	3.78	3.62	-4.2
80	2.66	2.64	-1.12
100	2.66	2.54	-4.5

From table 4.4, it is observed that at all CPU allocations, the percentage error is less than the 5% acceptable benchmark set initially in our experimental design. Besides the improvement in response time as CPU allocation increases, no observable trace of contention for resources in the experiment. As noted earlier, workload from unknown applications cannot be readily observed due to background processes but it can be estimated using poison as a good estimation of request arrival rate. It therefore infers that the model fits at the various CPU allocations of 20%, 40%, 60%, 80%, and 100%.

4.4 MODEL EVALUATION BASED ON PHP/APACHE SERVER

Consider sample data of the CPU allocation at 100% in the virtualized environment running PHP on APACHE as attached in Appendix C, the linear relationship was established using SPSS data analysis tool as shown in Table 4.5

Table 4.5 Sample Linear Regression Analysis of AppSton Application (PHP/Apache machine)

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	13.446	4.178		3.219	.002
	VAR00002	.952	.330	.384	2.882	.06

From the table above,it can be establish from second virtual machine application platform that$C_n = 13.45 + .952\lambda$ However, the linearity of the relationship between consumption and workload has not be ascertain since the significant difference is still greater than 0.05 as noted in the other two cases (table 4.1 and 4.2)

Similarly, comparison of observed and estimated mean response time as shown in figure 4.11 to 4.15

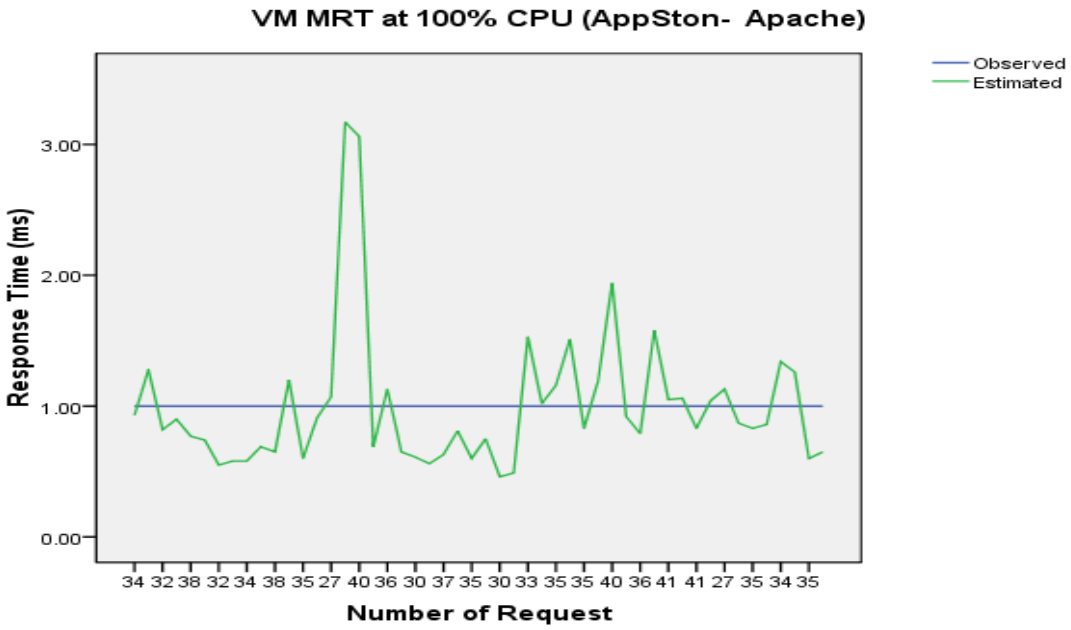


Fig 4.11: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 100% (PHP, Apache & MySQL)

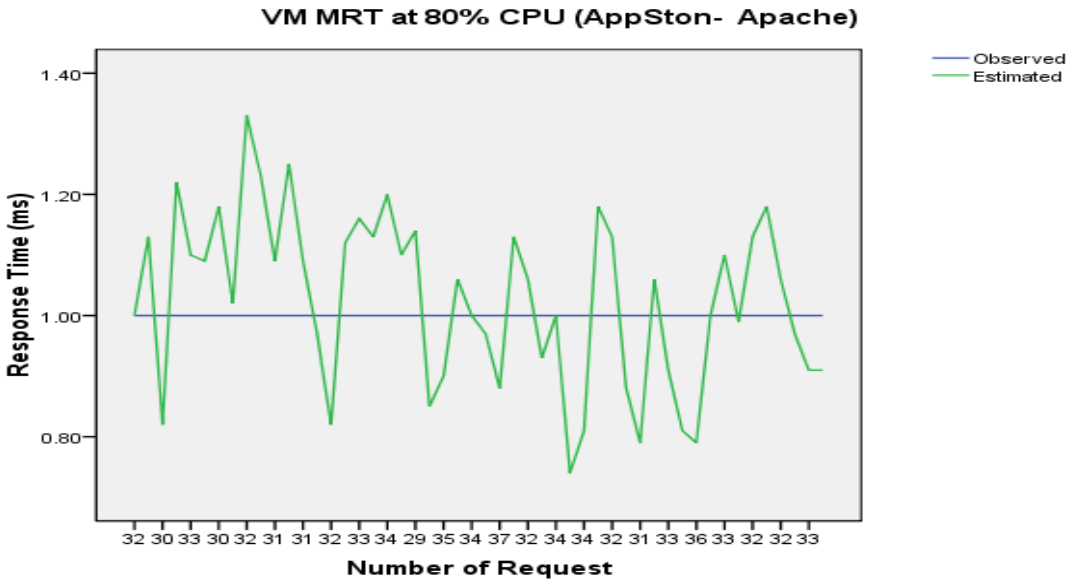


Fig 4.12: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 80% (PHP, Apache & MySQL)

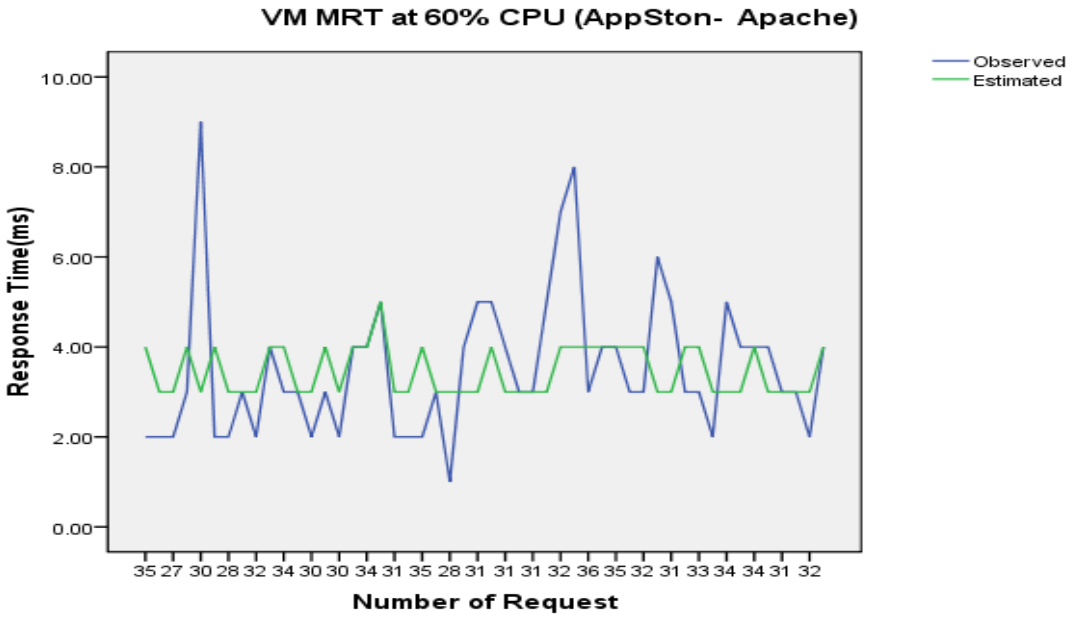


Fig 4.13: Performance Predictionof VM Mean Response Time (MRT) for CPU Allocation at 60% (PHP, Apache & MySQL)

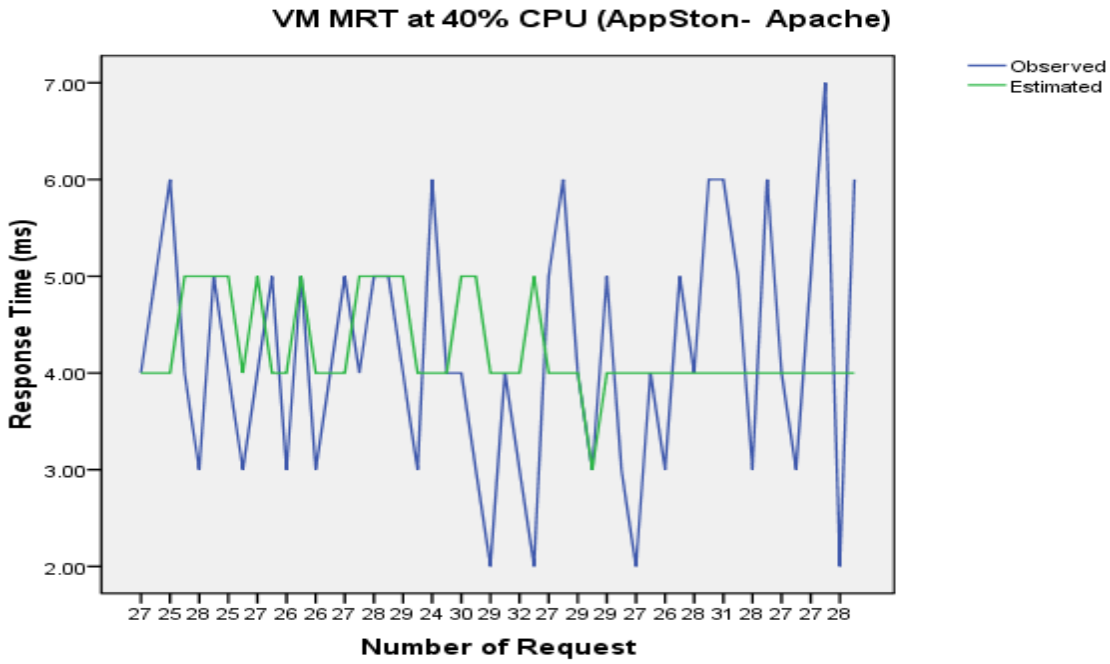


Fig 4.14: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 40% (PHP, Apache & MySQL)

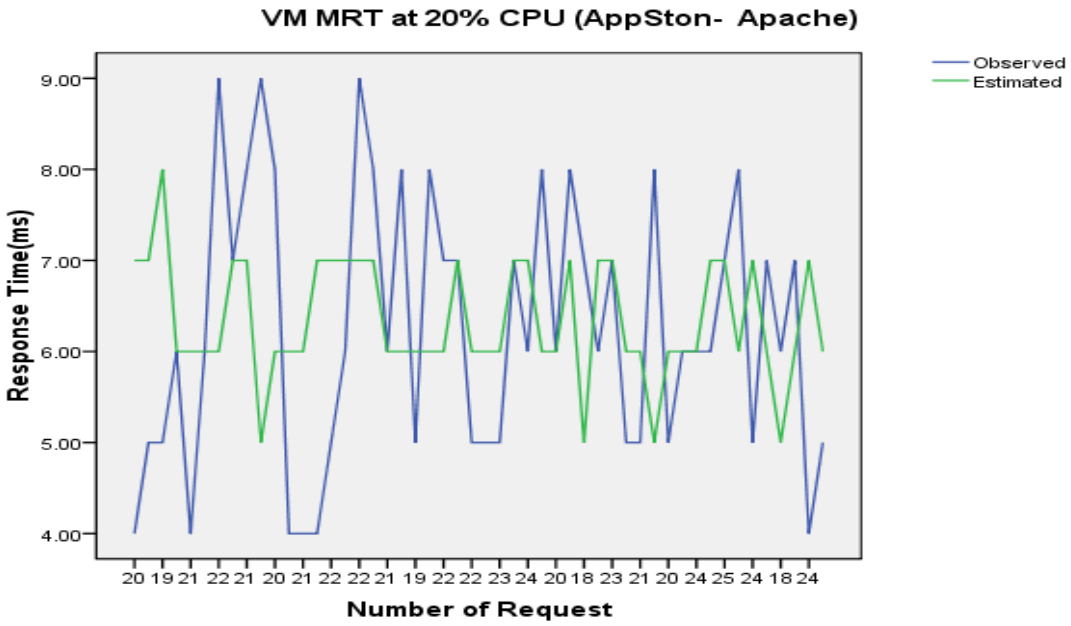


Fig 4.15: Performance Prediction of VM Mean Response Time (MRT) for CPU Allocation at 20% (PHP, Apache & MySQL)

From the figures above (Fig 4.11 - 4.15), in the experiment for AppSton application based on PHP/Apache and MySQL platform, it is observed that the performance model also estimated values that fits the observed/measured value as CPU allocation increases. The values are closely related at all CPU allocations of 20%, 40%, 60%, 80% and 100%.

To ascertain the validity of our model, consider the mean response time % error in the model as shown in the Table 4.6

Table 4.6 MRT Estimation Error for Apache/PHP application

% CPU Allocation	Avg Observed(s)	Avg Estimated (s)	% Error
20	6.24	6.32	1.2
40	4.18	4.30	2.8
60	3.52	3.50	-0.56
80	1	1.02	2.0
100	1	0.99	-1.0

The estimated values are higher when compared to the observed value at all CPU allocation. This is contrary to the case of exMaster where the estimated values are lower. However, in this experiment also, there is observable level of stability in the measured values at all CPU allocations.

Given a specific application and transaction/workload, the performance model predicted the CPU response time on the testbed and production environment. Consider the experiment using RUBiS standard workload, it is observed that as the number of threads increases, the response time also increases with CPU utilization. This observation shows that there is usually contention with low CPU allocation at 20% and high request rate from 300 to 500 as shown in See Appendix A.

On the other hand, the experiment using custom workload generator exMaster and AppSton based on ASP.Net/IIS and PHP/Apache technology respectively on the production server, it is observed that although response time increases with lower CPU allocations, as the CPU allocation is increased, the CPU consumption tends to stabilize even with increased workload and reducing response time as shown in Appendix B and C.

Comparison of the results with Zhikui *et al*(2009) with 2,400 transactions on the average per minute using the RUBiS workload shows that at all CPU allocations (20%, 40%, 60%, 80% and 100%), Zhikui's response time is considerably higher (doubled) when workload is lower. Zhikui's response time tends to reduce when the workload increases at same CPU allocation. This is due to the average number of transactions assumed during the time interval of measurement. The number of transactions is expected to increase as workload increases. Hence the base model is valid at low workload at all CPU allocations since at high workload it observed values differed significantly from the estimated values using the model.

Similarly, the linearity relationship between CPU consumption and workload as proposed by Zhikui *et al*has not been ascertained in the experiments since the significant difference in the estimation of parameters is greater than 0.05 in all the three experiments conducted. These result shows that CPU consumption does not depend only on workload but perhaps on other factors that cannot easily be measured in a virtualized environment.

4.5 ANSWERS TO RESEARCH QUESTION

To answer the research questions, we consider our research question.

i. Question 1: How application-level performance in terms of response time is affected in virtualized system.

Answer: From statistics gathered in the three (3) experiments (RUBiS, exMaster,AppSton) it is observed that at 60%, 80% and 100% CPU allocation, there is negligible difference in the CPU consumption as the workload/request increases. The consumption is much more stable with lowering response time. This implies that the application has reached its peak in CPU requirement. Consequently if the DRS is allowed to dynamically allocated resources subject to workload variance, application performance may be affected by the design and architecture of the underlying framework.

ii. Question 2: How can we efficiently manage server resources despite highly varying application workloads? (The data center environment makes these challenges particularly difficult since it requires solutions with high scalability and extreme speed to respond quickly to fluctuating internet workloads).

Answer: To effectively manage server resources with varying workload, it is important to understand the resource requirement of the application that will utilize the server resources.

Considering the offline data (Appendix A) in the case of RUBiS, variation in the CPU allocations improved the response time as the workload increases. Similarly considering the online data from real time production server (Appendix B & C), as workload increases response time tends to increase. However in the custom application experiment (exMaster &

AppSton), there is no observable trace of contention in result; the response time reduces as CPU allocation increases even with the difference in the number of request. The model helps to estimate response time (performance) at the various allocated CPU such that as allocation increases, the performance will improve even though consumption may also increase with the increasing workload, an optimal point can be reached using the model estimation.

iii. Question 3: What effect has resource management on application performance?(The CPU and memory allocated to a virtual machine can be dynamically adjusted and hence it is important to know the impact of this flexibility attribute on application performance).

Answer: From the values collected as well as those estimated by the proposed model, it is observed that the response time increases with low CPU allocations and high workload resulting in an increased CPU consumption/utilization. In the case of the VMware hypervisor, if the DRSis employed to dynamically utilize idle resources shared by the virtual machine on the host, it can reduce the impact of contention for resources when the workload reaches its peak.

iv. Question 4: To what extent can CPU contention affect application performance? (Due to the consolidation nature of data centers, resource contention is intensified and there is need to minimize the resources used for the different applications while satisfying service level objectives).

Answer: The allocated CPU in virtual machines has corresponding effect on the amount of CPU utilization/consumption. In the presence of intense workload and low CPU allocations as

observed in experiment 1 (RUBiS appendix A) at 20% and 40%, contention for CPU resources increases which in turn increases the response time thus affecting the performance.

Chapter Five:

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1INTRODUCTION

This chapter discusses the summary, conclusion and recommendations.

5.2SUMMARY

Empirical result (Appendix D) shows an improvement of Zhikui et al (2009). The researcher took into consideration the number of CPU cores available to virtual machines as an input parameter and removed the assumption of average number of transactions mix as workload benchmark. The following were also considered to test reliability of the proposed model and to address the problems identified in the base model.

i. The researcher concentrated on measuring performance both offline (on testbed) and online (on the production server) base on VMware Hypervisors with LinuxUbuntu 12.1 server and windows server 2008 asguest operating system with three (3) different applications as workload generator (RUBiS as standard workload, and exMaster & AppSton as custom workload for online production environment)

ii. The researcherconsidered the number of CPU cores available to the virtual machines in modeling application performance.

iii. The researcher also considered a consolidated approach in the experimental design by deploying all the tier on a single host. This is the core target of virtualization technology.

iv. The researcher considered the number of threads/request concurrently running as a measure of workload on the server.

v. Although the researcher aggregated request rate from tested application, the request from other hidden applications (Antivirus, firewall, etc) and consumptions from background applications or unknown processes has been taken into consideration.

vi. The researcher took into consideration a typical datacenter design practice.

vii. The researcher concentrated on measuring response time on the server to reduce uncertainty of challenges that affects performance like network availability and bandwidth, speed and configuration of client system, underlying technology of client system (For instance it's clearly observable that Google Chrome browser is faster than Mozilla Firefox when in page navigation whereas Mozilla Firefox is faster in downloads that Google Chrome). This approach is a trade-off for time spent waiting for CPU resources when measuring performance on the virtual machine.

Table 5.1 Comparison of base model and proposed model.

S/No	**Item**	**Zhikui's Model**	**Proposed Model**
1	Consolidation of server resources	Not consolidated	Consolidated
2	Measurement of model parameters	Offline	Offline and Online(production server)
3	Number of Application experiment as workload generator	One (1)	Three (3)
4	Experiment testbed	Simple architecture	Typical datacenter design
5	Hypervisor	Xen	Vmware
6	CPU allocation	Controlled allocation	Controlled allocation and Number of CPU cores available to VM
7	Hidden Transactions	Assumed to be implicit	Considered as unknown transaction to be estimated using poison as an approximation of request arrival

8	Measurement of response	On client	On the VM
9	Measure of workload	Transaction	Aggregate transaction/request

5.3 CONCLUSION

Consequently,the improved model was validated for the different CPU allocation values 20%, 40%, 60%, 80% and 100% for all the three experiments conducted. The response time increases as workload increases (See Appendix A, B and C) and it is also observed from the analysis of both results that the response time increases with low CPU allocation especially at 20% and 40% CPU allocations. Thus by varying the CPU allocation based on business needs, the proposed model can predict the performance of the VM given a particular workload.

5.3RECOMMENDATIONS

Researchersshould focus on studying the performance of virtual machines to ascertain the variation in the linearityrelationship between CPU consumption and workload which has proven otherwise from the analysis. Research should focus on complexities and other hidden factors associated with virtual machine performance as well as inclusion of memory in the modeling of performance in consolidated VM.

REFERENCES

Appleby K., Fakhouri S., Fong L., Goldszmid, M. Krishnakumar, S., Pazel, D.,Pershing, J., and Rochwerger, B. Oceano (2001) SLA-based management of a computingutility. In Proceedings of the IFIP/IEEE Symposium on Integrated Network Managementpp35-60

Apte Varsha, Rukma P. Verlekar, Bhavish Aggarwal and Prakhar Goyal (2007)AMethodology and Tool for Performance Analysis of Application Hosting Centers , in MASCOTS'07, the 15th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Istanbul. pp 207

BenevenutoFabrícioand José Renato Santos (2006)Performance Models for Virtualized Applications, *Lecture Notes in Computer Science, Volume 4331, Frontiers of High Performance Computing and Networking; ISPA 2006* Workshops pp.427-439, 2006

Bodik P., Sutton C., Fox A., Patterson D., and Jordan M. (2007), Response-Time Modeling for Resource Allocation andEnergy-Informed SLAs. In *Workshop on Statistical Learning Techniques for Solving Systems Problems(MLSys)*, Whistler, Canada.

Bugnion Edouard, Devine Scott, and Rosenblum Mendel.(1997) DISCO: *Running Commodity Operating Systems on Scalable Multiprocessors.SOSP* (1997), pp. 143–156.

Chandra Abhishek, Gong Weibo, and Shenoy Prashant.(2003)*Dynamic resource allocation for shared data centers using online measurements.* ACM SIGMETRICS international conference on Measurement and modelingof computer systems (San Diego, CA, USA, 2003), ACM, pp. 300–301.

C. Stewart, T. Kelly, and A. Zhang, (2007)Exploiting nonstationarity for performance prediction*ACM SIGOPS Operating Syst. Rev.*, vol. 41, no. 3

Chase, J., Anderson, D., Thakar, P., Vahdat, A., and Doyle, R. (2001) Managing energy and server resources in hosting centers. *18th ACM Symposium on Operating Systems Principles (SOSP)*p. 103116.

Chen Y., Iyer S, Sahai A., andMilojicic D. (2009)A Systematicand Practical Approach to Generating Policies from ServiceLevel Objectives. *11th IFIP/IEEE Int. Symposium on Integrated Network Management*,

Church, K., Hamilton, J., and Greenberg, A. (2008) Delivering embarrassingly distributedcloud services.*Hotnets Vol II*

Cohen I., Zhang S., Goldszmidt M., Symons J., Kelly T., andFox A., (2005)Capturing, indexing, clustering and retrievingsystem history, *20th ACM Symposium on Operating Systems Principles (SOSP)*,

Cong-FengJian and Jian Wan, (2011) Resource Allocation in Contending Virtualized Environment through Stochastic Virtual Machine Performance Modelling and Feedback, *Journal of Information Science and Engineering.*Vol 10 pp 50

DeCandia G.,Hastorun D.,Jampani M., Kakulapati G., Lakshman A., Pilchin A., Sivasubramanian S., Vosshall P., and Vogels W. (2007)Dynamo: Amazon's Highly Available Keyvalue Store, *ACM Symposium on Operating Systems Principles*, Stevenson, WA.

Garfinkel, Tal, Pfaff, Ben, Chow, Jim, Rosenblum, Mendel, and Boneh, Dan. Terra(2003) A virtual machine-based platform for trusted computing. *13th CM symposium on Operating systems principles* (Bolton Landing, NY,USA, 2003), ACM, pp. 193–206.

Grit, Laura, Irwin, David, ,Yumerefendi, Aydan, and Chase, Jeff.(2006) Virtual machinehosting for networked clusters: Building the foundations for autonomic orchestration.In:*Workshop on Virtualization Technology in Distributed Computing* (VTDC).

Gupta, D., Cherkasova, L., Gardner, R., and Vahdat, A(2006). Enforcing performanceisolation across virtual machines in xen. *ACM/IFIP/USENIX 7th International Middleware Conference* (Middleware'2006), Melbourne, Australia

Heo J., Henriksson D., Liu X., and Abdelzaher T.,(2007)Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study, *28th IEEE Int. Real-Time Syst. Symp. (RTSS)*.

Isard, Michael, Budiu, Mihai, Yu, Yuan, Birrell, Andrew, and Fetterly Dennis.(2007), Dryad: Distributed data-parallel programs from sequential building blocks. *2nd ACM SIGOPS/EuroSys European Conference on Computer Systems,Lisbon, Portugal*, ACM, pp. 59–72.

Iyer Ravi, Illikkal Ramesh, Tickoo Omesh, Zhao Li,and Apparao Padma(2009) Measuring, Modeling and Managing VM Shared Resources in Computer Networks. In:*The International Journal of Computer and Telecommunications Networking*, Volume 53 Issue 17 http://dx.doi.org/10.1016/j.comnet.2009.04.015

Kephart J. O, Chan H., Das R., Levine D., Tesauro G., Rawson F., and Lefurgy C. (2007)Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs, *4th IEEE Int. Conf. Autonomic Computing (ICAC)*, Washington, DC, USA,

Krishnamurthy D., Rolia J. A., and Majumdar S.(2006), A synthetic workload generation technique for stress testing session-based systems.*IEEE Trans. Software Engineering*, 32(11), pp 868–882

Kumar V., Schwann K., Iyer S., Chen Y., and Sahai A.(2008), AState Space Approach to SLA based Management. *In Proceedings.of the IEEE/IFIP NOMS*

Lim, K., Ranganathan, P., Chang, J., Patel, C., Mudge, T., and Reinhardt, S.(2008) Understanding and designing new server architectures for emerging warehouse-computingenvironments Computer Architecture, *2008.ISCA'08. 35th International Symposium*, pp. 315–326.

Liu X., Zhu X., Padala P., Wang Z., andSinghal S.(2007) Optimal multivariate control fordifferentiated services on a shared hostingplatform.*46th IEEE Conference on Decision and Control.*

Lu, Pin, and Shen, Kai.(2007) Virtual machine memory access tracing with hypervisor exclusive cache.*Usenix Annual Technical Conference* (June 2007).

Lublin U and Feitelson D,(2003)The workload on parallel supercomputers: modeling the characteristics of rigid jobs,*Journal of Parallel and Distributed Computing*, Vol 13 pp14

Mark B. Friedman; (2012) Measuring Processor Utilization in Windows and Windows applications*http://demandtech.com/wp-content/uploads/2012/04/Measuring-Processor-Utilization-in-Windows.pdf*

Nelson Michael, Lim Beng-Hong, and Hutchins, Greg.Fast (2005) Transparent Migration for Virtual Machines,*USENIX Annual Technical Conference* (2005).

Padala P., Zhu X., Uysal M., Wang Z., Singhal S.,Merchant A., Salem K., and Shin K (2007). Adaptive control of virtualized resources in utilitycomputing environments. In *Proceedings of EuroSys2007*, March 2007.

Pinheiro, Eduardo, Weber, Wolf-Dietrich, and Barroso, LuizAndr´e. (2007) Failure trendsin a large disk drive population,*5th USENIX conference on File and Storage Technologies* (Berkeley, CA, USA, 2007), USENIX Association,pp. 2–2.

PradeepPadala, Kai-Yuan Hou, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant (2009): Automated control of multiple virtualized resources.*EuroSys'09, April 1–3, 2009*, Nuremberg, Germany

Price T., A Note on the Effect of the Central ProcessorService Time Distribution on Processor Utilization inMultiprogrammed Computer Systems,*J. ACM*, Vol. 23,No. 2, pp. 342—346, 1976.

Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., and Zhu, X.(2008) No powerstruggles: A unified multi-level power management architecture for the data center*ASPLOS*, March (2008).

Rao J., Bu X.,Xu Z., Wang L., and Yin G.Vconf:(2009)A reinforcement learning approach to virtual machines auto-configuration,*ICAC '09:Proceedings of the 6th international conference on Autonomic computing*.

Ricado Lent (2011), Evaluating the Performance and Power Consumption of Systems with Virtual Machines. *3rd IEEE International Conference on cloud computing and technological science*.

Rolia J., Cherkasova L., Arlitt M., and Andrzejak A., (2005) A capacity management service for resource pools,*5th International Workshop Software Performance (WOSP)*, Palma, IllesBalears, Spain, pp. 229-237.

Ruth, Paul, Rhee, Junghwan, Xu, Dongyan, Kennell, Rick, and Goasguen, Sebastien (2006) Autonomic Live Adaptation of Virtual Computational Environments ina Multi-Domain Infrastructure. *IEEE International Conference on Autonomic Computing (ICAC)*

Stewart C., Kelly T., and Zhang A.,(2007)ExploitingNonstationarity for Performance Prediction.In *Proceedings of EuroSys* 2007

Stewart, Christopher, Kelly, Terence, Zhang, Alex, and Shen, Kai.(2008)A dollar from15 cents: cross-platform management for internet services.*ATC'08: USENIX2008 Annual Technical Conference on Annual Technical Conference* (Boston, Massachusetts,2008), pp. 199–212.

Urgaonkar B., Pacifici G., Shenoy P., Spreitzer M., and Tantawi A., (2005)An Analytical Model for Multi-tier InternetServices and its Applications. In:*Proceedings of ACMSIGMETRICS*, June 2005

Urgaonkar, B., Rosenberg, A., and Shenoy, P. (2007) Application placement on a cluster ofservers, *International Journal of Foundations of Computer Science*, vol. 18, pp. 1023–1041.

Villela D, Pradhan P., and Rubenstein D. (2004)Provisioning Servers in the Application Tier for E-commerce Systems,*12th IWQOS*, June 2004.

Virtamo J (2003) Processor sharing Queue, *38 3141 Teletrafic theory*, pp 1-11

VMmark, Features of VMmark, Virtualization Benchmark, *http://www.vmware.com/products/vmmark/features.html*

Vmware.com, (2006) Virtualization Overview White Paper, *http://www.vmware.com/pdf/virtualization.pdf*,

Vugt Van Sander (2012), Memory and CPU allocation in Xen environments: Optimizing performance. Accessed on November 21, 2012. Available at: *http://searchservervirtualization.techtarget.com/tip/Memory-and-CPU-allocation-in-xen- environments-Optimizing-performance*

Wang X and Wang Y. (2009) Co-con: Coordinatedcontrol of power and application performance forvirtualized server clusters. *IWQOS '09: Proceedings of the 17th Int'l Workshop on Qualityof Service*, Charleston, SC, USA, 2009.

Wester, Benjamin, Cowling, James, Nightingale, Edmund B., Chen, Peter M., Flinn,Jason, and Liskov, Barbara.(2009) Tolerating latency in replicated state machines throughclient speculation. In:*Proceedings of NSDI* (Berkeley, CA, USA, 2009).

Wood T., Cherkasova L., Ozonat K. M., and Shenoy P. J. (2008). Profiling and modeling resource usage of virtualized applications". In Valerie Issarny and Richard E. Schantz, editors, *Middleware*, volume 5346 of Lecture Notes in Computer Science, pages 366–387.

XiaoqiaoMeng, CanturkIsci, (2008), Efficient Resource Provisioning in Computer Clouds via VM Multiplexing,*IBM T. J. Watson Research Center*,Hawthorne.

Zhang, Q., Cherkasova, L., and Smirni, E.(2007), A Regression-Based analytic model for dynamic resource provisioning of Multi-Tier applications,.In Proceedings. *ICAC* (2007).

Zhao, Weiming, and Wang, Zhenlin (2009), Dynamic memory balancing for virtual machines *2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (Washington, DC, USA, 2009), ACM,pp. 21–30.

Zhikui Wang, and YuanChen, Daniel Gmach, Sharad Singhal, Brian J. Watson,Wilson Rivera, Xiaoyun Zhu, and Chris D. Hyser (2009), Application level performance in virtualized server environment, *IEEE Transaction on Network and service management*, Vol 6, No 4.

Appendix A: Performance Matrix for virtual machinesbased on standard workload (RUBiS)

20% CPU				40% CPU				60% CPU				80% CPU				100% CPU			
		Response Time				Resonse time				Resonse time				Resonse time				Resonse time	
AVR Req	DB Utiliz	Obs.	Est	AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est
100	69	25	27.82	100	56	18	15.91	100	50	17	12.50	100	9	3	1.24	100	12	2	1.70
100	71	30	30.60	100	57	23	16.57	100	48	17	11.54	100	10	3	1.39	100	14	3	2.03
100	68	24	26.56	100	54	20	14.67	100	53	16	14.10	100	9	2	1.24	100	12	2	1.70
100	69	30	27.82	100	55	28	15.28	100	54	22	14.67	100	10	2	1.39	100	15	3	2.21
100	71	32	30.60	100	64	31	22.22	100	53	24	14.10	100	11	3	1.54	100	14	2	2.03
100	64	18	22.22	100	65	27	23.21	100	49	10	12.01	100	11	3	1.54	100	11	2	1.54
100	70	24	29.17	100	66	26	24.26	100	47	16	11.08	100	12	3	1.70	100	12	2	1.70
100	71	22	30.60	100	57	19	16.57	100	46	14	10.65	100	11	3	1.54	100	13	2	1.87
100	70	28	29.17	100	60	23	18.75	100	46	20	10.65	100	14	3	2.03	100	12	3	1.70
100	69	27	27.82	100	56	22	15.91	100	47	19	11.08	100	14	3	2.03	100	13	3	1.87
200	69	54	55.65	200	60	34	37.50	200	57	18	33.14	200	14	4	4.07	200	12	4	3.41
200	68	56	53.13	200	60	35	37.50	200	46	18	21.30	200	15	4	4.41	200	11	3	3.09
200	71	53	61.21	200	62	38	40.79	200	46	15	21.30	200	17	5	5.12	200	12	4	3.41
200	71	55	61.21	200	66	48	48.53	200	47	18	22.17	200	15	4	4.41	200	12	4	3.41
200	74	70	71.15	200	64	45	44.44	200	48	17	23.08	200	12	5	3.41	200	12	4	3.41

200	73	65	67.59	200	60	40	37.50	200	46	22	21.30	200	14	4	4.07	200	12	5	3.41
200	71	58	61.21	200	57	37	33.14	200	47	24	22.17	200	17	4	5.12	200	12	4	3.41
200	68	58	53.13	200	58	34	34.52	200	49	20	24.02	200	16	3	4.76	200	13	3	3.74
200	69	55	55.65	200	57	36	33.14	200	45	19	20.45	200	17	5	5.12	200	11	4	3.09
200	71	60	61.21	200	56	35	31.82	200	47	18	22.17	200	16	5	4.76	200	13	4	3.74
300	70	88	87.50	300	48	34	34.62	300	48	29	34.62	300	12	4	5.11	300	14	5	6.10
300	69	87	83.47	300	61	42	58.65	300	51	34	39.03	300	15	5	6.62	300	12	3	5.11
300	70	83	87.50	300	64	57	66.67	300	47	30	33.25	300	12	4	5.11	300	12	3	5.11
300	70	87	87.50	300	64	58	66.67	300	48	34	34.62	300	14	6	6.10	300	14	6	6.10
300	71	88	91.81	300	63	59	63.85	300	50	37	37.50	300	14	4	6.10	300	12	5	5.11
300	69	88	83.47	300	65	68	69.64	300	51	38	39.03	300	12	3	5.11	300	13	6	5.60
300	69	87	83.47	300	58	49	51.79	300	52	39	40.63	300	11	3	4.63	300	11	5	4.63
300	71	86	91.81	300	56	44	47.73	300	49	36	36.03	300	13	3	5.60	300	11	4	4.63
300	69	86	83.47	300	59	54	53.96	300	48	36	34.62	300	13	3	5.60	300	11	5	4.63
300	69	88	83.47	300	62	54	61.18	300	51	36	39.03	300	13	3	5.60	300	11	4	4.63
400	69	100	111.29	400	60	70	75.00	400	50	47	50.00	400	14	3	8.14	400	13	7	7.47
400	69	115	111.29	400	61	76	78.21	400	49	48	48.04	400	14	4	8.14	400	13	8	7.47
400	71	120	122.41	400	61	81	78.21	400	50	47	50.00	400	12	7	6.82	400	11	7	6.18
400	70	132	116.67	400	64	87	88.89	400	51	50	52.04	400	14	7	8.14	400	14	8	8.14
400	70	123	116.67	400	63	83	85.14	400	49	47	48.04	400	13	8	7.47	400	15	8	8.82

400	68	123	106.25	400	63	84	85.14	400	48	46	46.15	400	13	8	7.47	400	10	6	5.56
400	68	126	106.25	400	60	81	75.00	400	49	47	48.04	400	12	7	6.82	400	15	8	8.82
400	68	129	106.25	400	59	77	71.95	400	47	43	44.34	400	14	8	8.14	400	13	7	7.47
400	70	120	116.67	400	58	71	69.05	400	47	44	44.34	400	13	8	7.47	400	12	8	6.82
400	68	103	106.25	400	61	69	78.21	400	46	43	42.59	400	11	8	6.18	400	13	7	7.47
500	66	118	121.32	500	41	67	43.43	500	51	60	65.05	500	10	8	6.94	500	14	7	10.17
500	68	129	132.81	500	63	98	106.42	500	52	67	67.71	500	14	10	10.17	500	11	8	7.72
500	71	128	153.02	500	62	109	101.97	500	55	70	76.39	500	16	10	11.90	500	10	7	6.94
500	69	127	139.11	500	63	111	106.42	500	54	74	73.37	500	17	13	12.80	500	14	9	10.17
500	68	130	132.81	500	65	120	116.07	500	53	68	70.48	500	17	11	12.80	500	13	10	9.34
500	71	135	153.02	500	64	119	111.11	500	55	76	76.39	500	18	14	13.72	500	11	8	7.72
500	70	132	145.83	500	66	119	121.32	500	57	78	82.85	500	16	12	11.90	500	12	9	8.52
500	69	140	139.11	500	64	113	111.11	500	56	79	79.55	500	14	12	10.17	500	13	10	9.34
500	71	134	153.02	500	63	100	106.42	500	57	79	82.85	500	14	14	10.17	500	12	9	8.52
500	70	141	145.83	500	64	101	111.11	500	58	87	86.31	500	15	11	11.03	500	12	10	8.52

Appendix B: Performance Matrix for virtual machines based on Custom workload (exMaster – ASP.Net/IIS)

20% CPU				40% CPU				60% CPU				80% CPU				100% CPU			
		Response Time				Resonse time				Resonse time				Resonse time				Resonse time	
AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est	AVR Req	DB Utiliz	Obs	Est
10	93	18	17	12	85	10	9	14	67	4	4	17	59	3	3	18	53	3	3
9	93	17	15	13	87	13	11	18	60	3	3	22	50	3	3	23	52	4	3
11	94	19	22	13	83	5	8	15	66	3	4	15	55	2	2	15	54	2	2
10	94	18	20	12	85	7	9	18	65	3	4	24	54	1	4	24	54	3	4
11	93	21	18	10	90	13	11	14	67	3	4	21	56	7	3	21	56	3	3
12	92	15	17	9	91	10	11	11	66	4	3	11	56	2	2	10	55	2	2
12	91	18	15	11	91	12	14	15	65	2	3	15	57	3	2	15	53	2	2
13	91	19	16	10	87	11	8	14	63	3	3	14	56	1	2	14	52	2	2
12	90	16	14	11	90	14	12	17	65	3	4	17	40	2	1	17	53	3	2
12	90	17	14	12	86	10	9	21	64	2	5	22	45	2	2	20	53	2	3
14	96	28	42	11	87	11	9	21	67	4	5	21	45	3	2	21	45	3	2
12	92	18	17	12	90	16	14	18	66	3	4	24	48	3	3	24	43	2	2
12	92	18	17	14	87	12	12	19	62	4	4	18	52	4	2	18	46	1	2
12	90	19	14	12	85	11	9	17	61	6	3	12	52	3	2	12	55	2	2
15	91	20	19	9	82	8	5	17	68	4	5	12	50	2	2	10	45	2	1
12	93	21	20	8	90	8	9	16	65	4	4	18	54	1	3	21	50	3	3
12	94	23	24	9	87	3	8	19	63	3	4	21	54	3	3	23	51	1	3
11	93	21	18	13	86	11	10	18	64	3	4	19	54	2	3	20	49	3	2
10	96	27	30	14	87	12	12	17	64	5	4	19	54	4	3	19	46	3	2
12	95	25	29	14	86	12	11	16	63	4	3	18	54	3	3	18	45	2	2
11	92	18	16	15	89	13	15	13	65	2	3	13	53	1	2	13	47	3	1
13	92	19	19	12	86	9	9	16	65	4	4	16	53	2	2	16	46	2	2
14	91	18	18	12	86	11	9	15	65	2	3	20	53	3	3	20	45	1	2
11	90	14	12	14	84	11	9	13	66	3	3	23	52	4	3	23	47	2	3
10	89	13	10	12	84	5	8	18	64	5	4	18	54	2	3	18	45	3	2

12	90	13	14	12	82	6	7	13	63	2	3	23	54	2	3	23	54	3	3
11	87	11	9	12	81	9	6	17	64	4	4	13	56	3	2	13	54	2	2
12	88	15	11	15	86	11	12	16	64	6	4	15	55	3	2	15	55	3	2
13	89	18	13	12	86	9	9	14	64	8	3	17	53	2	2	16	56	4	3
13	91	21	16	12	83	5	7	12	63	3	3	19	52	2	3	20	56	3	3
12	94	20	24	11	84	5	7	17	68	8	5	19	53	2	3	18	56	3	3
13	89	14	13	10	82	9	6	14	67	4	4	14	53	1	2	14	57	3	2
12	90	13	14	12	81	6	6	15	65	2	3	19	54	3	3	18	56	4	3
11	91	14	14	10	89	12	10	14	64	3	3	18	54	2	3	21	56	2	3
10	92	12	14	9	87	7	8	15	65	6	3	19	54	2	3	23	56	2	4
13	85	8	9	11	83	6	7	16	65	5	4	17	55	5	3	19	58	4	3
12	87	9	10	12	82	6	7	17	65	2	4	20	56	3	3	19	54	4	3
12	90	16	14	14	87	11	12	14	67	3	4	21	56	2	3	21	54	3	3
12	91	16	15	13	83	13	8	18	64	5	4	24	56	3	4	24	54	3	4
11	90	14	12	12	81	5	6	17	63	4	4	19	57	2	3	17	54	3	2
14	88	14	13	12	80	9	6	18	63	5	4	23	56	2	4	24	55	4	4
12	89	16	12	13	84	17	9	17	64	5	4	22	56	2	4	21	56	2	3
11	91	13	14	12	86	9	9	17	63	4	4	18	56	4	3	10	54	2	1
12	92	19	17	14	87	9	12	18	64	4	4	13	58	3	2	17	56	3	3
12	89	18	12	12	87	11	10	15	63	6	3	18	54	3	3	18	52	2	2
11	87	11	9	12	89	10	12	15	63	3	3	19	54	3	3	19	51	2	2
13	92	16	19	12	86	8	9	12	65	2	3	18	54	4	3	17	53	4	2
12	90	16	14	15	84	11	10	13	67	2	3	18	53	3	3	21	54	4	3
10	87	9	8	12	83	6	7	16	67	3	4	18	56	3	3	18	54	3	3
12	90	17	14	12	84	9	8	12	64	4	3	15	54	3	2	21	55	2	3

Appendix C: Performance Matrix for virtual machines based on Custom workload (AppSton – PHP/Apache)

20% CPU					40% CPU				60% CPU				80% CPU				100% CPU			
AVR Request	DB Utilization		Response Time		AVR Request	DB Utilization	Resonse time		AVR Request	DB Utilization	Resonse time		AVR Request	DB Utilization	Resonse time		AVR Request	DB Utilization	Resonse time	
			Observed	Estimated			Observed	Estimated			Observed	Estimated			Observed	Estimated			Observed	Estimated
20	73	3	4	7	27	55	4	4	35	49	2	4	32	20	1	1.00	34	18	1	0.93
22	71	4	5	7	27	57	5	4	30	48	2	3	32	22	1	1.13	41	20	1	1.28
19	76	4	5	8	25	53	6	4	27	50	2	3	30	18	1	0.82	32	17	1	0.82
20	69	5	6	6	34	55	4	5	30	49	3	4	39	20	1	1.22	41	15	1	0.90
21	71	3	4	6	28	60	3	5	30	48	9	3	33	21	1	1.10	38	14	1	0.77
21	69	5	6	6	26	61	5	5	31	49	2	4	31	22	1	1.09	27	18	1	0.74
22	70	8	9	6	25	63	4	5	28	46	2	3	30	24	1	1.18	32	12	1	0.55
22	71	6	7	7	24	57	3	4	27	47	3	3	29	22	1	1.02	31	13	1	0.58
21	73	7	8	7	27	60	4	5	32	46	2	3	32	25	1	1.33	34	12	1	0.58
18	69	8	9	5	28	56	5	4	34	47	4	4	33	23	1	1.23	37	13	1	0.69
20	69	7	8	6	26	57	3	4	34	46	3	4	31	22	1	1.09	38	12	1	0.65
21	68	3	4	6	25	60	5	5	31	45	3	3	30	25	1	1.25	41	19	1	1.20
21	70	3	4	6	26	57	3	4	30	46	2	3	31	22	1	1.09	35	12	1	0.60
22	71	3	4	7	26	55	4	4	35	46	3	4	31	20	1	0.97	29	20	1	0.91
22	73	4	5	7	27	52	5	4	30	46	2	3	32	17	1	0.82	27	24	1	1.07
22	73	5	6	7	25	60	4	5	34	46	4	4	30	23	1	1.12	38	40	1	3.17
22	71	8	9	7	28	57	5	5	34	46	4	4	33	22	1	1.16	40	38	1	3.06
22	73	7	8	7	29	56	5	5	40	49	5	5	34	21	1	1.13	37	13	1	0.69
21	69	5	6	6	29	57	4	5	31	45	2	3	34	22	1	1.20	36	20	1	1.13
20	71	7	8	6	28	56	3	4	31	47	2	3	33	21	1	1.10	35	13	1	0.65
19	70	4	5	6	24	59	6	4	35	48	2	4	29	24	1	1.14	30	14	1	0.61

21	69	7	8	6	26	56	4	4	32	46	3	3	31	18	1	0.85	33	12	1	0.56
22	70	6	7	6	30	56	4	5	28	46	1	3	35	17	1	0.90	37	12	1	0.63
23	70	6	7	7	31	54	3	5	28	48	4	3	36	19	1	1.06	40	14	1	0.81
22	68	4	5	6	29	54	2	4	31	46	5	3	34	19	1	1.00	35	12	1	0.60
22	69	4	5	6	33	52	4	4	33	47	5	4	38	17	1	0.97	40	13	1	0.75
23	69	4	5	6	32	51	3	4	31	45	4	3	37	16	1	0.88	30	11	1	0.46
25	70	6	7	7	29	56	2	5	29	45	3	3	34	21	1	1.13	32	11	1	0.49
24	69	5	6	7	27	56	5	4	31	45	3	3	32	21	1	1.06	33	27	1	1.53
23	68	7	8	6	29	53	6	4	32	45	5	3	34	18	1	0.93	37	18	1	1.02
20	69	5	6	6	29	54	4	4	32	47	7	4	34	19	1	1.00	35	21	1	1.16
23	71	7	8	7	24	52	3	3	34	47	8	4	29	17	1	0.74	31	28	1	1.51
18	70	6	7	5	29	51	5	4	36	45	3	4	34	16	1	0.81	35	16	1	0.83
22	71	5	6	7	25	59	3	4	36	48	4	4	30	24	1	1.18	38	20	1	1.19
23	70	6	7	7	27	57	2	4	35	49	4	4	32	22	1	1.13	40	28	1	1.94
23	68	4	5	6	27	53	4	4	36	44	3	4	32	18	1	0.88	36	17	1	0.92
21	68	4	5	6	26	52	3	4	32	49	3	4	31	17	1	0.79	36	15	1	0.79
18	68	7	8	5	25	57	5	4	31	47	6	3	30	22	1	1.06	38	25	1	1.58
20	69	4	5	6	28	53	4	4	31	46	5	3	33	18	1	0.91	41	17	1	1.05
24	68	5	6	6	29	51	6	4	32	47	3	4	34	16	1	0.81	34	20	1	1.06
24	67	5	6	6	31	50	6	4	33	48	3	4	36	15	1	0.79	41	14	1	0.83
26	68	5	6	7	29	54	5	4	34	45	2	3	34	19	1	1.00	38	18	1	1.04
25	70	6	7	7	28	56	3	4	34	44	5	3	33	21	1	1.10	27	25	1	1.13
23	68	7	8	6	23	57	6	4	33	43	4	3	28	22	1	0.99	34	17	1	0.87
24	69	4	5	7	27	57	4	4	34	47	4	4	32	22	1	1.13	35	16	1	0.83
21	71	6	7	6	25	59	3	4	31	45	4	3	30	24	1	1.18	36	16	1	0.86
18	70	5	6	5	27	56	5	4	31	46	3	3	32	21	1	1.06	34	24	1	1.34
21	69	6	7	6	28	54	7	4	31	47	3	3	33	19	1	0.97	38	21	1	1.26
24	71	3	4	7	28	53	2	4	32	46	2	3	33	18	1	0.91	35	12	1	0.60
22	70	4	5	6	26	54	6	4	33	46	4	4	31	19	1	0.91	38	12	1	0.65

Appendix D: Comparison of base model and the proposed model

20% CPU				40% CPU				60% CPU				80% CPU				100% CPU			
AVR Request	DB Utilization	Response Time		AVR Request	DB Utilization	Resonse time		AVR Request	DB Utilization	Resonse time		AVR Request	DB Utilization	Resonse time		AVR Request	DB Utilization	Resonse time	
		Estimated	Khikui			Estimated	Khikui			Estimated	Khikui			Estimated	Khikui			Estimated	Khikui
100	69	27.82	53.42	100	56	15.91	30.55	100	50	12.50	24.00	100	9	1.24	2.37	100	12	1.70	3.27
100	71	30.60	58.76	100	57	16.57	31.81	100	48	11.54	22.15	100	10	1.39	2.67	100	14	2.03	3.91
100	68	26.56	51.00	100	54	14.67	28.17	100	53	14.10	27.06	100	9	1.24	2.37	100	12	1.70	3.27
100	69	27.82	53.42	100	55	15.28	29.33	100	54	14.67	28.17	100	10	1.39	2.67	100	15	2.21	4.24
100	71	30.60	58.76	100	64	22.22	42.67	100	53	14.10	27.06	100	11	1.54	2.97	100	14	2.03	3.91
100	64	22.22	42.67	100	65	23.21	44.57	100	49	12.01	23.06	100	11	1.54	2.97	100	11	1.54	2.97
100	70	29.17	56.00	100	66	24.26	46.59	100	47	11.08	21.28	100	12	1.70	3.27	100	12	1.70	3.27
100	71	30.60	58.76	100	57	16.57	31.81	100	46	10.65	20.44	100	11	1.54	2.97	100	13	1.87	3.59
100	70	29.17	56.00	100	60	18.75	36.00	100	46	10.65	20.44	100	14	2.03	3.91	100	12	1.70	3.27
100	69	27.82	53.42	100	56	15.91	30.55	100	47	11.08	21.28	100	14	2.03	3.91	100	13	1.87	3.59
200	69	55.65	26.71	200	60	37.50	18.00	200	57	33.14	15.91	200	14	4.07	1.95	200	12	3.41	1.64
200	68	53.13	25.50	200	60	37.50	18.00	200	46	21.30	10.22	200	15	4.41	2.12	200	11	3.09	1.48
200	71	61.21	29.38	200	62	40.79	19.58	200	46	21.30	10.22	200	17	5.12	2.46	200	12	3.41	1.64
200	71	61.21	29.	200	66	48.53	23.	200	47	22.17	10.	200	15	4.41	2.1	200	12	3.41	1.6

			38				29				64				2				4
200	74	71.15	34.15	200	64	44.44	21.33	200	48	23.08	11.08	200	12	3.41	1.64	200	12	3.41	1.64
200	73	67.59	32.44	200	60	37.50	18.00	200	46	21.30	10.22	200	14	4.07	1.95	200	12	3.41	1.64
200	71	61.21	29.38	200	57	33.14	15.91	200	47	22.17	10.64	200	17	5.12	2.46	200	12	3.41	1.64
200	68	53.13	25.50	200	58	34.52	16.57	200	49	24.02	11.53	200	16	4.76	2.29	200	13	3.74	1.79
200	69	55.65	26.71	200	57	33.14	15.91	200	45	20.45	9.82	200	17	5.12	2.46	200	11	3.09	1.48
200	71	61.21	29.38	200	56	31.82	15.27	200	47	22.17	10.64	200	16	4.76	2.29	200	13	3.74	1.79
300	70	87.50	18.67	300	48	34.62	7.38	300	48	34.62	7.38	300	12	5.11	1.09	300	14	6.10	1.30
300	69	83.47	17.81	300	61	58.65	12.51	300	51	39.03	8.33	300	15	6.62	1.41	300	12	5.11	1.09
300	70	87.50	18.67	300	64	66.67	14.22	300	47	33.25	7.09	300	12	5.11	1.09	300	12	5.11	1.09
300	70	87.50	18.67	300	64	66.67	14.22	300	48	34.62	7.38	300	14	6.10	1.30	300	14	6.10	1.30
300	71	91.81	19.59	300	63	63.85	13.62	300	50	37.50	8.00	300	14	6.10	1.30	300	12	5.11	1.09
300	69	83.47	17.81	300	65	69.64	14.86	300	51	39.03	8.33	300	12	5.11	1.09	300	13	5.60	1.20
300	69	83.47	17.81	300	58	51.79	11.05	300	52	40.63	8.67	300	11	4.63	0.99	300	11	4.63	0.99
300	71	91.81	19.59	300	56	47.73	10.18	300	49	36.03	7.69	300	13	5.60	1.20	300	11	4.63	0.99
300	69	83.47	17.81	300	59	53.96	11.51	300	48	34.62	7.38	300	13	5.60	1.20	300	11	4.63	0.99
300	69	83.47	17.81	300	62	61.18	13.05	300	51	39.03	8.33	300	13	5.60	1.20	300	11	4.63	0.99
400	69	111.29	13.35	400	60	75.00	9.00	400	50	50.00	6.00	400	14	8.14	0.98	400	13	7.47	0.90

400	69	111.29	13.35	400	61	78.21	9.38	400	49	48.04	5.76	400	14	8.14	0.98	400	13	7.47	0.90
400	71	122.41	14.69	400	61	78.21	9.38	400	50	50.00	6.00	400	12	6.82	0.82	400	11	6.18	0.74
400	70	116.67	14.00	400	64	88.89	10.67	400	51	52.04	6.24	400	14	8.14	0.98	400	14	8.14	0.98
400	70	116.67	14.00	400	63	85.14	10.22	400	49	48.04	5.76	400	13	7.47	0.90	400	15	8.82	1.06
400	68	106.25	12.75	400	63	85.14	10.22	400	48	46.15	5.54	400	13	7.47	0.90	400	10	5.56	0.67
400	68	106.25	12.75	400	60	75.00	9.00	400	49	48.04	5.76	400	12	6.82	0.82	400	15	8.82	1.06
400	68	106.25	12.75	400	59	71.95	8.63	400	47	44.34	5.32	400	14	8.14	0.98	400	13	7.47	0.90
400	70	116.67	14.00	400	58	69.05	8.29	400	47	44.34	5.32	400	13	7.47	0.90	400	12	6.82	0.82
400	68	106.25	12.75	400	61	78.21	9.38	400	46	42.59	5.11	400	11	6.18	0.74	400	13	7.47	0.90
500	66	121.32	9.32	500	41	43.43	3.34	500	51	65.05	5.00	500	10	6.94	0.53	500	14	10.17	0.78
500	68	132.81	10.20	500	63	106.42	8.17	500	52	67.71	5.20	500	14	10.17	0.78	500	11	7.72	0.59
500	71	153.02	11.75	500	62	101.97	7.83	500	55	76.39	5.87	500	16	11.90	0.91	500	10	6.94	0.53
500	69	139.11	10.68	500	63	106.42	8.17	500	54	73.37	5.63	500	17	12.80	0.98	500	14	10.17	0.78
500	68	132.81	10.20	500	65	116.07	8.91	500	53	70.48	5.41	500	17	12.80	0.98	500	13	9.34	0.72
500	71	153.02	11.75	500	64	111.11	8.53	500	55	76.39	5.87	500	18	13.72	1.05	500	11	7.72	0.59
500	70	145.83	11.20	500	66	121.32	9.32	500	57	82.85	6.36	500	16	11.90	0.91	500	12	8.52	0.65
500	69	139.11	10.68	500	64	111.11	8.53	500	56	79.55	6.11	500	14	10.17	0.78	500	13	9.34	0.72
500	71	153.0	11.	500	63	106.4	8.1	500	57	82.85	6.3	500	14	10.17	0.7	500	12	8.52	0.6

		2	75			2	7				6				8				5
500	70	145.83	11.20	500	64	111.11	8.53	500	58	86.31	6.63	500	15	11.03	0.85	500	12	8.52	0.65