Innovative Systems Design and Engineering
ISSN 2222-1727 (Paper) ISSN 2222-2871 (Online)
Vol.7, No.8, 2016

Performance Modelling of Consolidated Virtual Machines

Egbunu I. Banya[1] Egbunu O. Charity[2] John T. Ogbiti[3]
1.Directorate of Info and Comm. Tech, University of Agriculture, Makurdi, Benue State, Nigeria
2.Department of Mathematics/Computer Science, University of Agriculture, Makurdi, Benue State, Nigeria
3.Department of Mathematical Sciences, Kaduna State University, Kaduna State Nigeria

Abstract
With rapid and flexible resource provisioning of virtualization in data centers, problems of determining optimal virtual machine (VM) placements and dealing with virtualization overheads have emerged due to workload fluctuations and changing needs. These challenges have impacts on system performance.In this work, a performance model based on queuing theory, statistical methods and basic theories on system performance is proposed 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 research also examined the relationships between virtualized CPU allocation, CPU contention, and application response time to identify the influence of CPU allocation and how it affects system performance.Comparing estimated values with measured values, empirical result shows that the proposed model validated for all the CPU allocations in the experiments conducted. The response time increases as workload increases; it is also observed from the analysis that the response time increases with low CPU. Thus by varying the CPU allocation base on business needs, am optimal point can be reached such that the CPU can be efficiently managed.

1. Background of Study

As application portfolios expands to meet up with the target of making virtually all processes information technology (IT) based; budgets have tightened as well. The best approach is 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).

A recent focus on reducing the economic costs of 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).

Resource contention is intensified in virtualized environments due to the consolidation nature of virtualization. Hence, the problem is how to minimize the allocation of CPU server resource to an IT service (or application), while satisfying service level objectives. It is therefore necessary to formally understand the factors that meddle with application performance and virtualized resource allocation in order to avoid over-provisioning or over-loading of physical IT resources.

1.1 Research Motivation

The primary goal of the service level objectives in virtualized data center is to ensure appropriate performance of the corresponding IT service. Hence, the responsibility is to monitor user experience and 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 application workloads? This research attempts to provide solution to these issue.

According to (Zhikui *et al.,* 2009), application performance in virtual machine can be modelled using the equation below

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

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 resource demand to meet application-level performance requirement based 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) where CPU cores available can be shared among virtual machines.
iii. The design architecture in Zhikui *et al* does not mimic a typical datacenter.
iv. Approach failed to consider hidden request/transaction been process by the CPU especially

on windows guest operating system where Antivirus and other services are running.

v. Zhikui et al 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).

Hence, this research focuses on improving Zhikui's model that optimizes performance of applications in virtualized data center. This research intends to solve these challenges mentioned above.

2.0 Overview of VMware Hypervisors

Virtualization became more eminent as a result of its aim in improving resource utilization through server consolidation. 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. It serves as a cornerstone for their vSphere cloud computing platform, provides a host of capabilities not currently available with any other hypervisors. E.g High Availability, Distributed Resource Scheduling Distributed Power Management, Fault Tolerance, and Site Recovery Manager VMmark (2012).

2.1 Evaluating Performance in Virtualized Environment

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 prototype and validation of the methodology was done on a small virtualized testbed via the use of Xen virtualization environment.

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.

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.

Zhikui *et al* (2009) presented a performance model based on workload transaction history using Xen hypervisor and RUBiS. The model can predict performance of multi-tier applications running on virtualized servers with variable CPU entitlements. The modelling approach is non-intrusive in the sense that the process of model parameter identification requires no additional instrumentation and the data used in this approach is readily available from standard system and application monitoring.

3.0 Experimental Method

To model the performance of virtual machines, the researcher deploys a typical virtualized datacenter setup using VMware EXSI 5.1 Hypervisor on HP SL170s, with 32GB RAM, 2.93GHz 2 4-core Intel Xeon Processor, 2x1TB HDD internal storage with 12TB SAN storage 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 as 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.

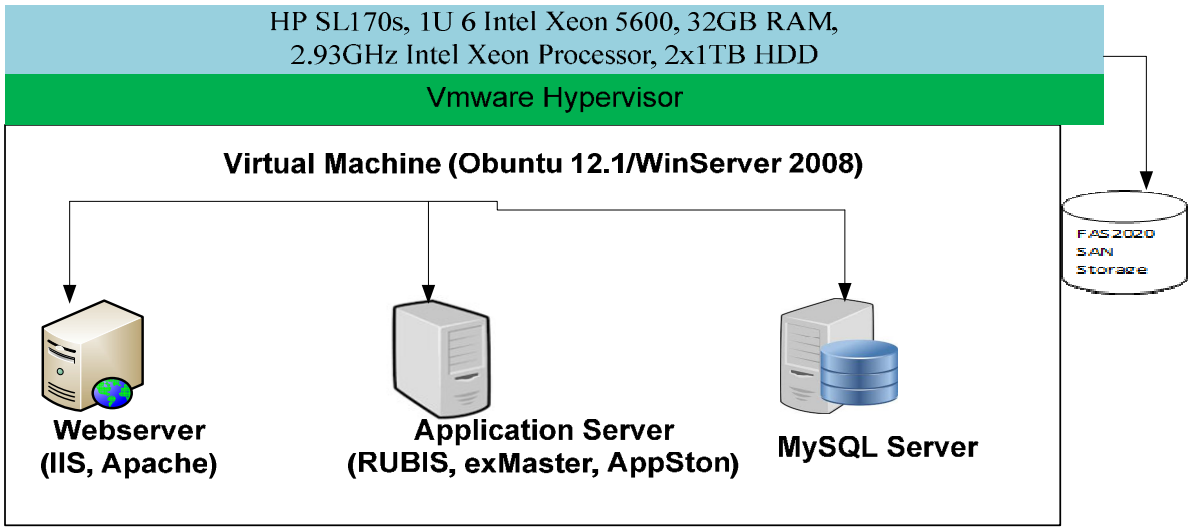

Fig 3.1: Experiment setup in virtualized Environment

Five (5) points of variation for CPU allocation to the virtual machines were considered as follows: 20%, 40%, 60%, 80% and 100%. At intervals of 60 seconds, the performance metrics were recorded. 50 measurements were taken for the different CPU allocations for the two core virtual machines (web and database server).

The PHP based RUBiS is a standard workload generator for offline parameter estimation. RUBiS is an open source benchmark application designed for several platforms (PHP, JAVA, etc) with MySQL as the database server and java client workload generator. Measurements were taken while varying the number of threads (100,200,300, 400 and 500) running on the client generating workload.

The exMaster is based on ASP.Net designed using C# and MySQL. exMaster is an online administrative result processing system designed to manage student academics records.

The AppSton is based on PHP/Apache and MySQL. AppSton is a comprehensive student portal designed to manage student records from admission, screening, payment of fees and registration of courses.

At peak periods when the various users of the custom applications (exMaster & AppSton) are busy, CPU allocations are varied and performance metrics are recorded. Workload estimation is based on the number of request 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.

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.

3.1 Assumptions

The accuracy of internal measurements of application response time is affected by the virtualization environment. In VMware, a virtual machine is transparent to the guest Operating System. Hence, measurements taken from inside the virtual machine guest are unaware of any delays introduced due to resource sharing. These delays are negligible due to the hypervisor design and non CPU resources are assumed to be adequately provisioned. The contention problem is reduced in VMware; this is handled by the dynamic resource scheduler (DRS) to load balance shared resources.

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).

3.2 Notations

Here are the notations and basic definitions used:

i. Resource allocation α to refer to the percentage of a physical resource capacity (e.g., CPU) that is allocated to a virtual machine. α=c/u
ii. Resource consumption c is the actual percentage of the physical resource consumed by a virtual machine during a given time interval.
iii. Resource utilization u=c/α.
iv. Ω Number of CPU cores available to virtual machine
v. n, N Number of virtual machines running separate tiers.
vi. i , I transactions 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

3.3 Model Generation

The approach employed here is to characterize response time as a probability distribution conditioned on CPU allocation and workload fluctuation 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 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*, the intensity of the workload can be defined as a vector ($\beta_1, \ldots, \beta_i$), where β_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 i can 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 δ_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, δ can 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 resources 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* 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 represented by $\frac{U_n}{1-U_n}$ where u_n is the CPU utilization of tier *n*.

According to Ricardo Lent (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 $0.9 < \sum_j \Omega(i,j) < 1$ 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

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

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

$$T_n = \frac{\gamma_n}{\Omega} \sum \frac{C_n}{\alpha_n - C_n} \qquad (3.6)$$

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.

$$C_n = \sum_{i=1}^{I} \lambda_{in}(\beta_i + \delta_i) \qquad (3.7)$$

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.

Since the transaction mix changes over time in a non-stationary way, and the relationship between C_n and β_i are linear, λ_{in} can be estimated through linear regression using (3.7)

4.0 Results and Discussions

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. The focus is to compare the estimated values and the measured values of the response time for the different CPU allocations.

4.1 Analyses based on RUBIS standard workload

To evaluate the performance model, we compare the values of the observed response time and the estimated response time as shown in the figures 4.1 to 4.5

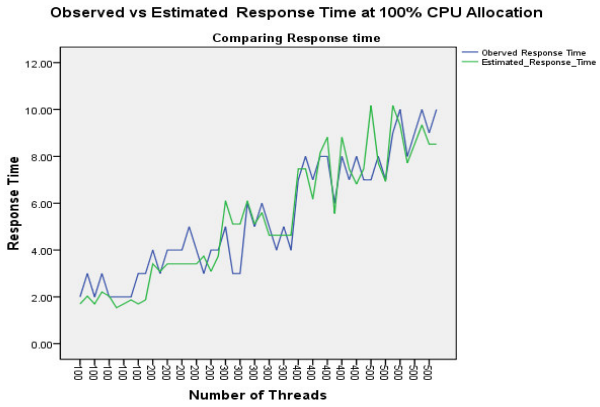

Fig 4.1: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 100%*

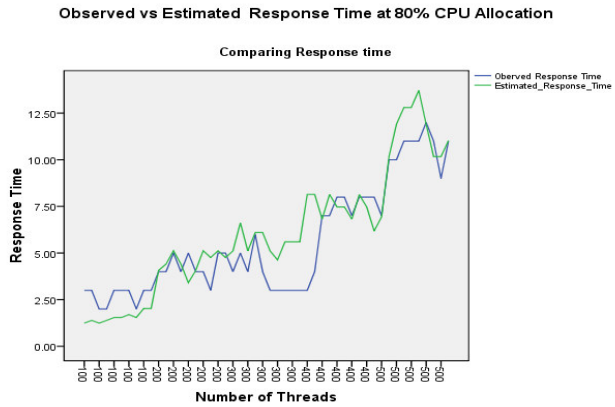

Fig 4.2: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 80%*

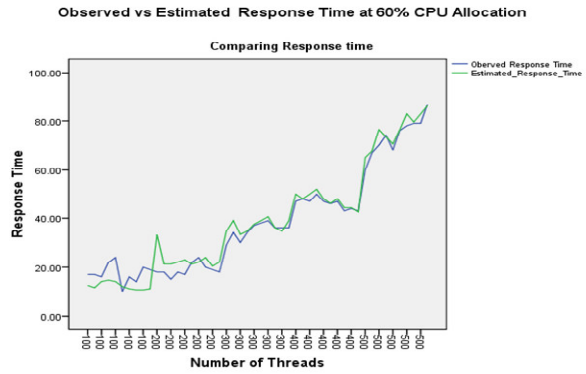

Fig 4.3: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 60%*

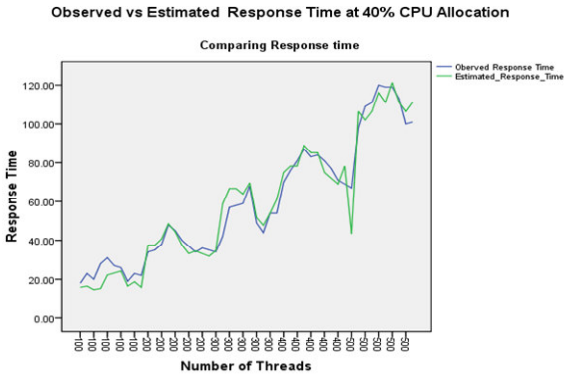

Fig 4.4: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 40%*

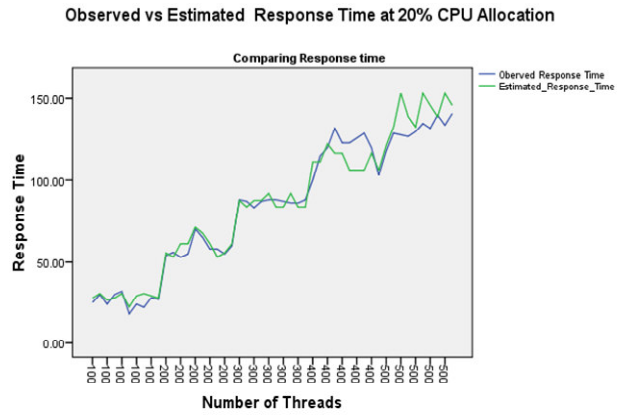

Fig 4.5: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 20%*

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 4.1

Table 4.1 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 4.1, 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% 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

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.

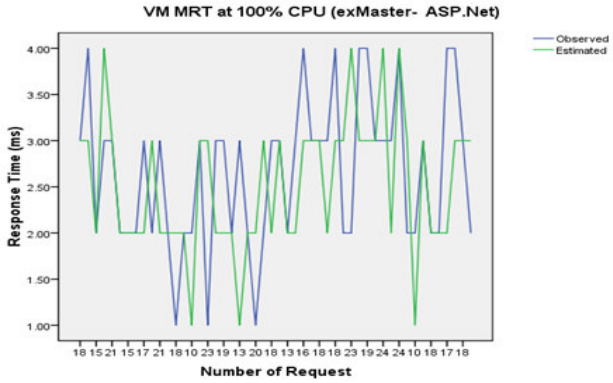

Fig 4.6: Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 100% (ASP.Net, IIS & MySQL)

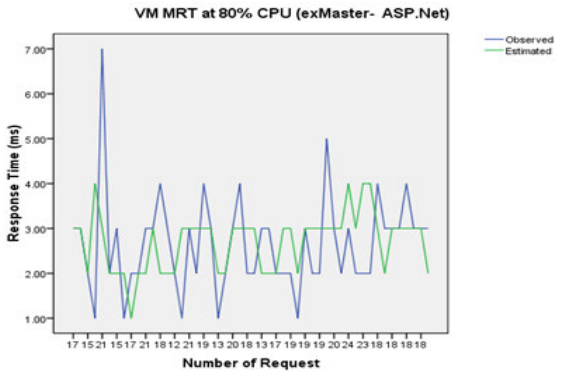

Fig 4.7: Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 80% (ASP.Net, IIS & MySQL)

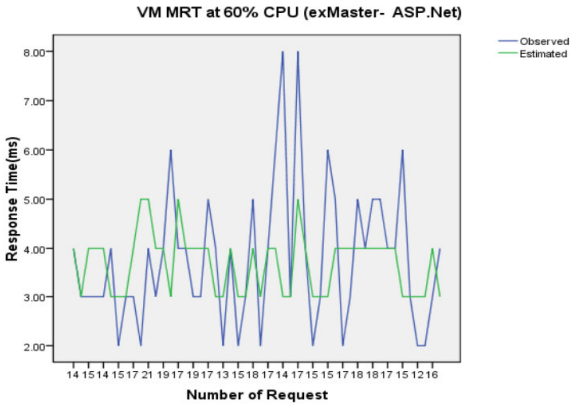

Fig 4.8: Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 60% (ASP.Net, IIS & MySQL)

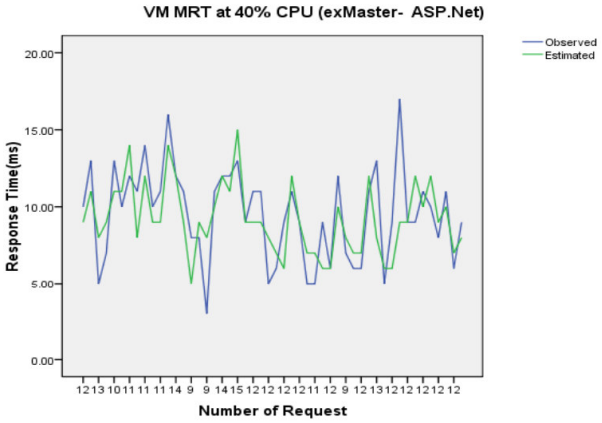

Fig 4.9: Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 40% (ASP.Net, IIS & MySQL)

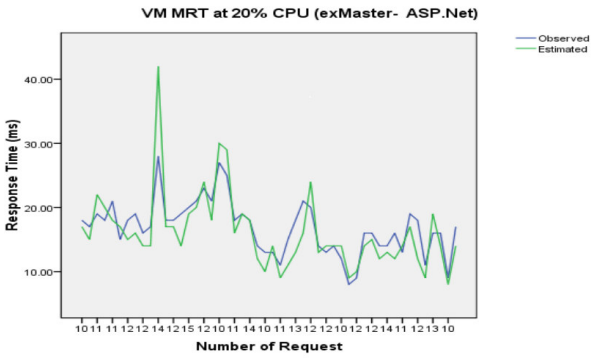

Fig 4.10: Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 20% (ASP.Net, IIS & MySQL)

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 4.3

Table 4.3 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 4.3, 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%.

4.3 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 4.4

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

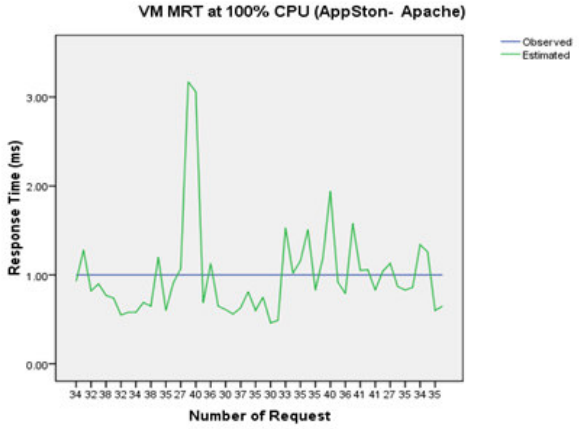

Fig 4.11: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 100% (PHP, Apache & MySQL)*

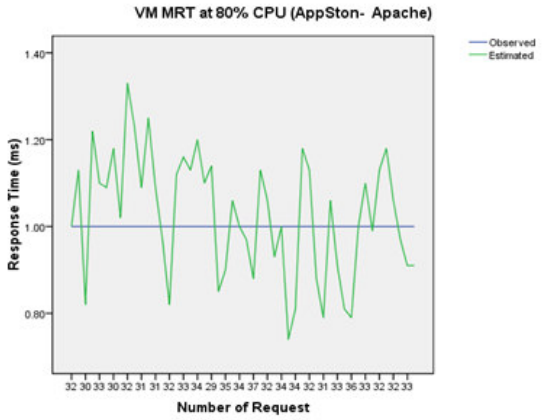

Fig 4.12: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 80% (PHP, Apache & MySQL)*

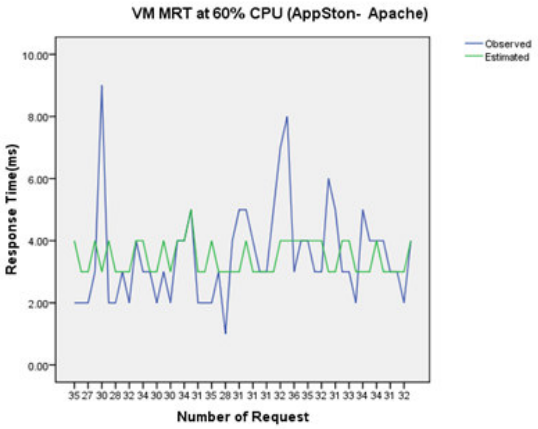

Fig 4.13: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 60% (PHP, Apache & MySQL)*

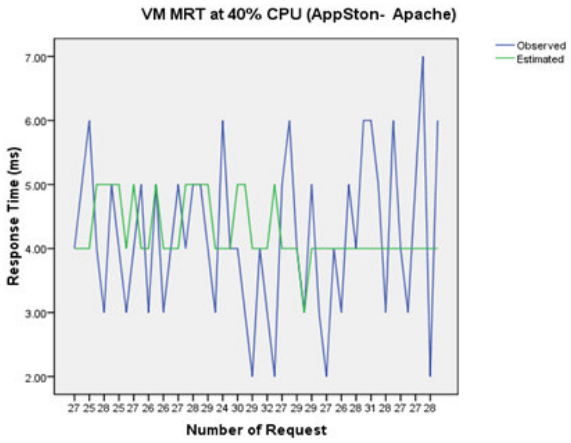

Fig 4.14: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 40% (PHP, Apache & MySQL)*

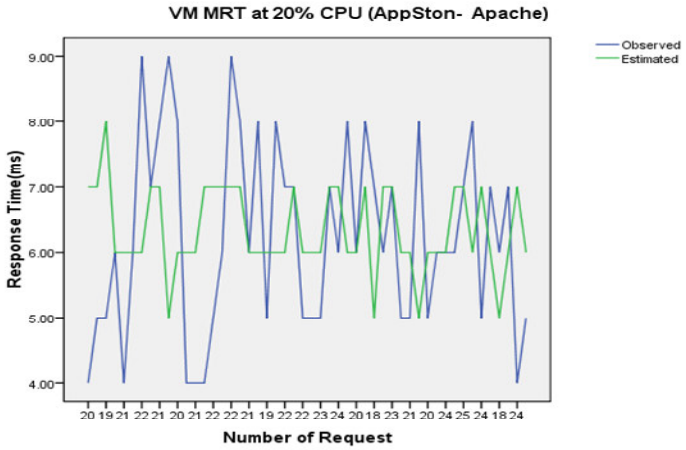

Fig 4.15: *Performance prediction of VM Mean Response Time (MRT) for CPU allocation at 20% (PHP, Apache & MySQL)*

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 4.5

Table 4.5 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.

5.0 Summary

In this work, the researcher presented a performance model based on queuing theory and statistical methods to model the performance of consolidated virtual machines hosted on VMware Hypervisors 5.0.

Although this model is an improvement on Zhikui's model which was based on Xen Hypervisor and Linux guest OS with transaction mix, the researcher took into consideration the following 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 Linux Ubuntu 12.1 server and windows server 2008 as guest 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 researcher considered 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.

Considering the RUBiS experiment, 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.

On the other hand, the experiment using custom workload generator exMaster and AppSton 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.

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 may be due to underlying hypervisor and 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.

5.1 Conclusion

Consequently, the proposed model was validated for the different CPU allocation all the experiments conducted. The response time increases as workload increases 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 base on business needs, am optimal point can be reached such that the CPU can be efficiently managed while satisfying service level objective. Contention arises from low CPU allocation and increases the response time for the end users.

The table below shows the comparison of the base model and the proposed model as well as the improvement.

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

It is worthy of note that there are other factors from the experiments that can affect the results of the model like the number of virtual machines or tiers, number of measurements considered, type of applications, hypervisor, guest OS as well as the interval for the collection of values, number of CPU cores available to virtual machine.

5.2 Recommendations

The researcher will continue to study the performance of virtual machines to ascertain the variation in the linearity relationship between CPU consumption and workload which has proven otherwise from the analysis. Research will focus on complexities and other hidden factors associated with virtual machine performance.

References

Apte Varsha, Rukma P. Verlekar, Bhavish Aggarwal and Prakhar Goyal (2007) A Methodology 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

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

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*

Pradeep Padala, 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

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*.

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*

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

Zhikui Wang, and Yuan Chen, 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.