International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.4, No.12, December 2015 DOI:10.15693/ijaist/2015.v4i12.149-169

LQR Based Optimal Control Scheduler for Optimizing Performance of Server Virtualization in Cloud Data Center Environment

Vedula Venkateswara Rao
Research Scholar/Department of CSE
Gitam Institute of Technology, Gitam University,
Visakhapatnam, India
Associate Professor/Department of CSE
Sri Vasavi Engineering College, Tadepalligudem

Dr.Mandapati.Venkateswara Rao
Professor/Department of IT
Gitam Institute of Technology, Gitam University,
Visakhapatnam, India

Abstract— **Data centers of today are rapidly moving towards the use of server virtualization as a preferred way of sharing a pool of server hardware resources between multiple 'guest domains' that host different applications. The hypervisors of the virtualized servers, such as the Xen use fair schedulers to schedule the guest domains, according to priorities or weights assigned to the domain by administrators. The hosted application's performance is sensitive to the scheduling parameters of the domain on which the application runs. However, the exact relationship between these parameters of the domain and the application performance measures such as response time or throughput is not obvious and not static as well. Furthermore, due to the dynamics present in the system there is need for continuous tuning of the scheduling parameters. The main contribution of our work is the design and implementation of a controller that optimizes the performance of applications running on guest domains. The main objective of our work is to devise a mechanism to dynamically set resource management parameters for the virtual machines in such a way that the specified goals are satisfied. We focus on a scenario where a specific target for the response time of an application may not be provided. The goal is to dynamically compute the CPU shares for the virtual machines in such a way that the application throughput should be maximized, while keeping the response time as low as possible, with the minimum possible allocation of CPU share for the guest domain. The optimizing controller design is based on the feedback control theoretic concept called Linear Quadratic Regulator (LQR). The controller computes the values of the scheduling parameters for every guest domain in such a way that it minimizes the CPU usage and response time, and maximizes throughput of the applications. To evaluate our work, we deployed multi-tier application in virtual machines hosted on the Xen virtual machine monitor. The performance evaluation results show that the controller brings the cap value close to the expected optimal value. The optimizing controller also rapidly responds to changes in the system when a disturbance task is introduced or load on the application is changed.**

Index terms -Cloud Computing, Data Center, Virtualization, hypervisor, Xen, virtual machine, Green IT, scheduler, Performance, Response Time, Throughput, Feedback Control Theory, Linear Quadratic Regulator, CAP Value, Weight Value, Optimized Value.

I. INTRODUCTION

The task of predicting & maintaining the system performance and capacity planning is becoming difficult due to increased complexity in the it applications and infrastructure. Service providers host the applications from different enterprise clients on the shared pool of hardware resources. Clients negotiate the service contract in the form of service level agreement (SLA) with service providers which depict all the related formal information about the contract and the performance guarantees. The performance guarantees include QoS (quality of service) requirements [6] [37] like desired response time or throughput of the application. Degraded performance leads to penalty cost due to SLA violation as well as dissatisfied clients which ultimately results in financial loss for the service providers. Over-provisioning of hardware resources has always been the easiest choice for service providers to avoid any performance problems. But it leads to inefficient and costlier resource management.

Cloud computing is a technology that numerous it organizations extend their hands in order to improve their financial ability. This is done by improving the various QoS parameters such as performance, throughput, reliability, scalability, load balancing, persistence, etc. The services such as disk storage, virtual servers, application design, development, testing environment are added advantages of the cloud computing technology. The cloud computing technology makes the resource as a single point of access to the client and is implemented as pay per usage [1][2]. Though there are various advantages in cloud computing such as prescribed and abstracted infrastructure, completely virtualized environment, equipped with dynamic infrastructure, pay per consumption, free of software and hardware installations, the major concern is the order in which the requests are satisfied. This evolves the scheduling of the resources. This allocation of resources must be made efficiently that maximizes the system utilization and overall performance. Cloud computing is sold on demand on the basis of time constrains basically specified in minutes or hours. Thus scheduling should be made in such a way that the resource should be utilized. Nowadays server virtualization[26][46] is heavily used to build it infrastructure is it allows sharing of resources among different applications while at the same time providing isolated environment called virtual machine for each application[7] [10] [13]. Virtual machine hosts an OS (operating system) in its secured isolated environment consisting of virtual CPU, main memory and IO devices. Virtual machine monitors (vmm) like VMware, Xen [26] [46] does the task of protection and resource allocation among individual virtual machines. Some of the benefits of server virtualization include consolidation of multiple OS on single physical server, live migration of a virtual machine from one physical server to another physical server. With these capabilities offered by server virtualization, managing a server farm becomes easier and cost effective. Sharing of the resources should not cause performance of an application adversely getting affected by the other applications running on the same hardware. Gupta et al[16] [17] describes the term performance isolation as the scenario in which performance of the client application remains same regardless of type and amount of workload of other applications sharing the resources. Performance isolation is an important goal in any shared hosting environment such as virtualized environment. Performance isolation can be achieved by properly allocating the resources among competing virtual machines [18] [27] [28] [31]. Vmm allocates the share of resources like CPU, main memory to each virtual machine [37]. For example, CPU scheduler in Xen [46] accepts two parameters named weight and cap for each of the virtual machine. Weight represents the relative share of a virtual machine, whereas cap represents the upper bound on CPU consumption by the virtual machine. Performance isolation can be achieved by setting the appropriate values of resource management parameters like weight and cap for each virtual machine. Dynamic nature of the workload should be considered while modelling the performance behavior of the applications residing in virtual machines. Client SLA’s keep on changing very frequently. Addition or removal of clients is also a continuous process. Same is the case with underlying hardware infrastructure which frequently gets scaled or upgraded with new hardware components. With these many sources of dynamics, delivering QOS to the applications hosted in the virtual machines becomes more complex. Our study focuses on devising a mechanism for computing the share of the resources to be allocated to each virtual machine in such a way that desired QoS is delivered to the applications running inside virtual machines.

In this paper, we are applying feedback control theory [35] to maintain the performance of the applications running inside virtual machines. Feedback control theory does online analysis of the system and attempts to maintain the output of the system around the desired values [17]. In virtualized environment scenario, output refers to the QoS requirements of the clients which need to get satisfied. Controller in a feedback system computes the values of input parameters which affect the working of the system which in turn affects the output delivered by the system. In virtualized environment, input parameters refer to the resource management parameters like main memory allocation to guest OS, or some scheduler specific parameters like weight, cap, time-slice for a guest OS.

II. EXISTING SYSTEM AND RELATED WORK

A. Background

Cloud computing is a recent technology and a lot of research are made in that domain to improve it. Also due to the relation between cloud and virtualization there are as well many researches on virtualization to enhance virtualization performances. Cloud computing is more and more popular and most of the enterprise begin to adopt it. However there are still some obstacles which can restrained the adoption of cloud services by enterprise such as the lack of standardization, reliability associate to the cloud, the security and so on. The reason of the adoption of cloud computing by enterprise is principally for economical reasons because cloud computing allow customers to reduce their hardware cost as well as energy consumption [1] and so on. Also there is no waste because customers only pay for what they are using. As seen previously there are many different type of virtualization. To be able to provide the best performances cloud computing is using para-virtualization as well as hardware-assisted virtualization. Full virtualization is not used in cloud computing due to poor performances cause by its considerable overhead. Virtualization technology is not a new technology however it has regain popularity in 2005 with the apparition of and Intel processors which had support for virtualization. Virtualization brings many advantages such as the improvement of security, the enhancement of the efficiency of server utilisation and so on. Also during the past few years due to the popularity of virtualization and its utilisation in the cloud computing many researchers have been made. From that research, lot of improvement has been made to try to obtain performances near to native performances.

B. Cloud Computing

Cloud computing is a new technology and evolve rapidly also it is difficult to match a good definition of cloud computing [1] [2] [5]. Because cloud computing is an evolving technology the definition is changes over the time. The U.S. Government's National Institute of Standards and Technology (NIST) tries to give an up to date definition of the cloud computing. The actual version of their definition is the version 15 in date of 10 July 2009 (Mell et al, 2009). According to the NIST cloud computing is on demand service which shares a pool of computer resources over a network. Cloud computing matches five essential characteristics which define the main functionalities provided by the cloud, three service models which give the level of service provided and four deployment models which indicate where the cloud is deployed and who can access to it. The main characteristics of the clouds are the following (Mell et al, 2009):

- On demand self-service: Users of the cloud can manage the resources in on demand basis and they only paid for what they consume.
- Broad network access: The resources provided by the cloud can be access by as any normal services through thin or thick clients such as laptop, PDA, mobile phones and so on.
- Resource pooling: The cloud provider serves pool of resources over multiple customers according to the demand. Client which access the service have no knowledge of the exact location of the cloud but may be able to provide a location at higher abstraction level such as country, state, data center and so on [18].
- Rapid elasticity: The resources provided by the cloud are highly scalable. Customer can rapidly scale up the resources that they need and then scale them down if there is no need to use it anymore. The scalability of the cloud gives a real modularity to the cloud. Also resources appear as infinite and customers have no need to make plan for provisioning (Armbrust et al, 2010) [28].
- Measured service: The resources provided by the cloud are controlled and optimized according to the resources capabilities. Also resources usage can be monitored control and reported to be able to provide transparency for both provider and consumer of the resources [32] [45].

C. Virtualization for Resource Sharing

Data centers of today are rapidly moving towards the use of server virtualization as a preferred way of sharing a pool of server hardware resources. The journey of virtualization technology started in 1960s when IBM first invented the concept of virtual machine to divide the computing power of mainframe servers into logical partitions. Virtual Machine Facility/370 better known as VM/370[10] and was one of the initial successful implementations of virtual machines by IBM which was based on their mainframe server IBM System/370. VM/370 had been in wide use inside IBM for mainly time-sharing purpose and operating system development. The emergence of virtual machines was due to expensive mainframe systems. Virtual machines provided a convenient way to share the mainframe among multiple users so as to effectively use the otherwise wasted resources. Later virtualization became unnecessary as inexpensive x86 based machines came into markets around 1980s and 1990s. Also the client-server model of the applications helped in building distributed model for computing which was cheaper than computing using mainframes. Then came the era of World Wide Web in late 1990s, where the computing needs started to increase exponentially. Around the same period, many organizations started the use of IT applications at massive scale for various operations. The under-utilized machines became major source of concern as the operational and management cost of the infrastructure was rising without actually leveraging the resources to significant extent. Many of the studies reported average use of the servers and desktop machines around 5-15%. This situation resulted in making a call to old virtualization technology in this era. In 1999 VMware[34] became the first company to release a virtualization product for x86 based machines which was named “VMware Virtual Platform”. At present, VMware server [34], VMware ESX [32], Xen Server [28], Microsoft Virtual server [34] are some of the popular server virtualization solutions available in the market. Server virtualization provides a way of sharing a resource pool between multiple guest domains that host different applications. An isolated execution environment called virtual machine (VM) which is also referred as a domain is provided. The virtual machine hosts an operating system (OS) which is provided with a virtual set of CPU, main memory and IO devices. Virtual machine monitor (VMM) is a software layer between these virtual machines and the hardware. VMMs carry out the task of protection, isolation and resource allocation among the individual virtual machines. Some of the benefits of adopting server virtualization include consolidation of multiple OS’s on a single physical server, pooling of the resources, uniform interface to the resource pool, and live migration of a virtual machine from one physical server to another physical server. With these and many more capabilities offered by server virtualization, managing a server farm becomes easier and cost effective.

D. Application Performance in Data Centers

The task of predicting and maintaining the system performance and doing capacity planning is becoming difficult due to increased complexity in the IT applications and infrastructure. Service providers host applications from different enterprise clients on a shared pool of hardware resources in data centers. Clients negotiate a service contract in the form of a Service Level Agreement (SLA) with service providers which include a description of the performance guarantees. The performance guarantees may include Quality of Service (QoS) requirements such as desired response time or throughput of the application. Degraded performance leads to SLA violation which results in penalty cost for the service providers[1]. It also results into dissatisfied clients which ultimately results in financial loss for the service providers. Over-provisioning of hardware resources has always been the easiest choice for service providers to avoid such performance problems. But it leads to inefficient resource management and costlier infrastructure. Resource allocation needs to be done dynamically so that shared resources can be reused among the application more effectively.

One interesting situation arises when there are no pre-specified desired values of performance metrics. The clients may not specify the desired values; instead they require the maximized performance at minimal cost. For example, response time of an application decreases with increase in CPU capacity with certain rate for some range of capacity. This rate starts to drop after certain CPU capacity. So utilizing more CPU does not yield performance at the same rate, hence the cost to benefit ratio goes up.

E. Application Performance in Virtual Machines

The performance of an application should not get adversely affected by the other applications running on the same hardware. Gupta et al [12] described the term performance isolation as the scenario in which performance of an application remains the same regardless of type and amount of the workload of other applications sharing the same resources. Performance isolation is an important goal in any shared hosting environment such as a virtualized environment. As we have seen in the example of the previous section, CPU capacity allocated to the application has a major impact on the performance of the application. To achieve performance isolation, appropriate resource allocation need to be done among the competing virtual machines. For deciding the resource shares for an application we need to understand how the resource scheduling process works in VMMs. A VMM allocates the share of resources such as CPU, main memory to each virtual machine. For example, the CPU scheduler in Xen named credit scheduler accepts two parameters weight and cap for each virtual machine. Weight represents the relative share of a virtual machine, whereas cap represents the upper bound on CPU consumption by a virtual machine. The value of cap puts the limit on CPU usage by a virtual machine. If sum of cap of all virtual machines running on the given CPU is less than the CPU capacity then CPU remains idle even if there is some runnable work present in the system. The performance of a hosted application is sensitive to the weight or cap given to the domain on which the application is running. However, the exact relationship between the value of the weight or cap of the domain, and the application performance metrics such as response time or throughput is not obvious. Therefore, determining the appropriate parameter values that would provide a certain QoS for an application is a difficult problem. To make things worse, there are many sources of dynamics which makes the task of delivering QoS to the applications hosted in the virtual machines much more complex. e.g. the dynamic nature of the workload, or changing client SLAs. Addition or removal of clients is also a continuous process. This is also the case with underlying hardware infrastructure which frequently gets scaled or upgraded with new hardware components. With all this dynamics, the exact relationship between application performance and the amount of resource allocated to the application is not so obvious and is not static. From this scenario we infer that performance isolation can only be achieved by monitoring the running system and tuning the appropriate values dynamically.

F. Feedback Control Theory

Feedback control has been in the history much longer than the virtualization. One of the known initial applications of feedback control can be found in windmills of 17th centuries [35]. The very famous invention of James Watt, the steam engine [35] had a centrifugal governor to control over-speeding of the mover. Another legendary example is of control mechanism in first controlled human flight by Wright brothers [35]. Some of the widely used applications of feedback control theoretic approach involves automobile cruise control, aircraft cruise control, temperature maintenance using thermostat[29][17][35]. A feedback control system monitors the values of output metrics of the system, processes it and computes the new values of input parameters to be set. These input parameters should be some configuration parameters of the system which have influence on the working of the system. Thus, setting the value of input parameter to a new value can result in change in the output. As there is this interdependency between input and output of the system, it is called as feedback system. An important feature of the feedback control system is that it does online analysis of the system and responds to changes in the system dynamically. Feedback control system design can be done in two steps. In the first step, the mathematical model of the system is constructed which relates the output to its past values and to the past as well as present values of input parameters. From the constructed system model, a most important part of feedback control system named controller is designed. The controller computes the values of input parameters to be set. A typical feedback control system takes the input called reference input which specifies the objective for control. This input may or may not be present in every case. If system accepts the reference input, the controller tries to compute the values of input parameters in such a way that the output delivered will be equal to the reference input. In some scenarios there is no reference input provided to the system. In such scenarios, the objective for feedback control is to tune the input parameters in such a way that certain metrics are optimized.

These metrics may include the values of some output or input parameters. A feedback control system also consists of other components which monitor and process the values of the output metrics of the system. The output in the context of applications running inside the virtual machines refers to the QoS requirements such as response time and throughput, where as the input parameters can be comprised of resource management parameters such as CPU scheduling parameters of VMs, or main memory allocation to the VMs. In this work, we focused only on CPU sharing. The CPU scheduling parameter weight is the relative share of a VM whereas the value of cap is the absolute limit on CPU consumption of a VM. As the value of cap provides direct control over the CPU usage by a VM, we are using the cap of VM as the input parameter to be tuned.

III. PROBLEM DESCRIPTION

This section starts with describing the basics of the virtualization. Subsequently we discuss the performance issues occurring in the virtualized environment. Then we define the problem statement

A. Virtualization

The term virtualization refers to the abstraction of resources. The user or the software process is not aware of the actual characteristics of the resource. Rather, they get a view of resource which is more familiar to them or which is more manageable by them. Our concern over here is about server/software virtualization which is more popularly known as virtual machine environment. Figure 1 shows a virtualized environment. Let us see some of the basic terms in server virtualization.

- Virtual Machine (VM): this is a virtual environment created by vmm (described below), which simulates all the hardware resources needed by an operating system. The OS running in such environment is called a guest OS. Guest OS has a virtual view of the underlying hardware.
- Virtual Machine Monitor (VMM/HYPERVISOR): this is the interface between the guest OS and the underlying hardware. All the administrative tasks like adding a new guest OS, allocation of resources to each of guest OS is done through vmm. Some popular examples of vmm are vmware [46], Xen [47]. In our study, we have used open source vmm solution Xen [25] [26] [47].
- Host OS: the native OS running on the given hardware is called the host OS. The vmm is installed on host OS. This OS has all the privileges on the given hardware.

In simpler terms we can describe the virtualization as follows. The actual physical resources are divided into logical partitions. Each of the logical partition is allocated to some guest OS. Each guest OS runs independently on a given partition. For host OS, guest OS's are like the normal processes running on it. The vmm interface is available in host OS through which guest OS's are managed.

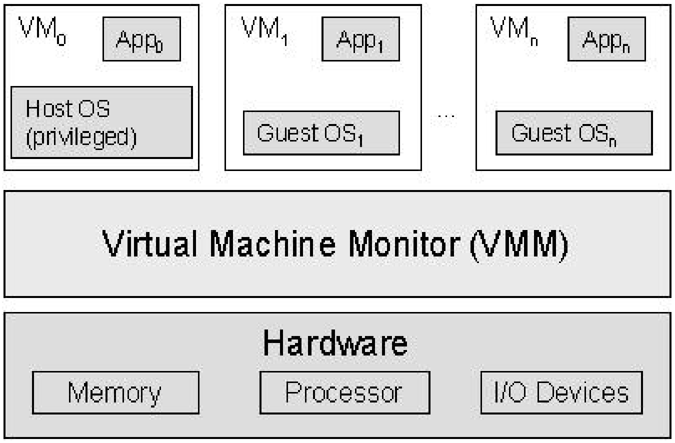


Figure 1. Virtualized Environment

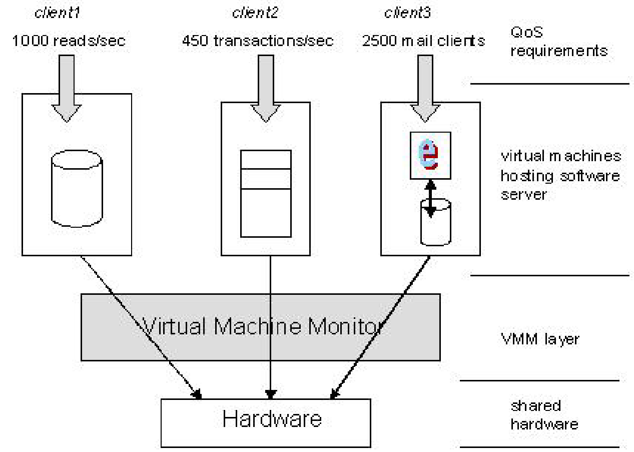
B. *Scheduling of Virtual Machines*

There are number alternatives for CPU scheduling in Xen like Borrowed Virtual Time (BVT), Simple Earliest Deadline First (SEDF) and Credit scheduler [5] which schedule the virtual machines on available set of processors. The latest scheduler for Xen is credit scheduler which is a proportional fair share SMP (Symmetric multiprocessor) scheduler. Each domain (including host OS) is assigned with number of virtual CPUs (VCPU), weight and cap values. Weight denotes share of a domain and is directly proportional to CPU requirement of a domain. The cap specifies the maximum amount of CPU a domain will be able to consume even if there is idle CPU. Thus credit scheduler works in non-work conserving mode when sum of cap of all domains is less than available CPU capacity. Each CPU manages a local run queue of runnable VCPUs sorted by VCPU priority. A VCPU's priority can be over or under depending upon whether that VCPU has exceeded its fair share of CPU in the ongoing accounting period. Accounting thread computes how many credits each virtual machine has earned and re computes the credits. Until a VCPU consumes its allotted credits, priority of VCPU is under. Scheduling decision is taken when a VCPU blocks or completes its time slice which is 30ms by default. On each CPU, the next VCPU to run is picked up from head of the run queue. When a CPU doesn't find a VCPU of priority under on its local run queue, it looks on other CPUs for VCPU with priority under. This load balancing mechanism guarantees each domain receives its fair share of CPU. No CPU remains idle when there is runnable work in the system.

C*. Performance Isolation and Application QoS*

In a virtualized environment, multiple software servers are hosted together on a single shared platform. Each server may belong to different owner. For each server, the QoS requirements are expressed by the client through Service Level Agreement (SLA) with the service provider. The task of the service provider is to maintain the performance such that SLA of any of the client does not get violated. SLA violations have pre-specified penalty costs associated with them. QoS crosstalk [20] occurs in a situation when maintaining QoS for some client results into degraded QoS for another client. Performance guarantees for the applications running inside the virtual machines can be fulfilled only if there is performance isolation across virtual machines. Figure 2 pictorially depicts the scenario of virtual machine environment.

Figure 2. Applications running inside virtualized Environment



Performance Isolation as described by [16] [17] is as follows: "Resource consumption by any of the virtual machines should not affect the promised performance guarantees to other virtual machines running on the same hardware" . Over-provisioning of the resources can be simplest solution to achieve performance isolation but then the whole essence of using virtualization can be lost. The ultimate aim is actually to increase the benefit of the service provider through better resource utilization with constraint of delivering QoS for each of the client. Hence some better solution other than over-provisioning is required. Let us see one example which describes this problem. In an earlier study we have shown that the behavior of the applications running inside the virtual machines remains unpredictable when there is IO load running on at least one virtual machine.

Effect on Mixed workload on the performance of applications in virtualized Environment in table 1.

<table>
<tr><td colspan="7">Statistics of web server running in virtualized environment</td></tr>
<tr><td></td><td>Weight</td><td>CAP</td><td>Load</td><td>CPU usage</td><td>Requests per sec</td><td>Transfer rate (Kbytes per sec)</td></tr>
<tr><td colspan="7">Experiment1: With Web Server running</td></tr>
<tr><td>Domain0</td><td>256</td><td>400</td><td>-</td><td>-</td><td colspan="2">NA</td></tr>
<tr><td>VM2</td><td>256</td><td>400</td><td>-</td><td>-</td><td colspan="2">NA</td></tr>
<tr><td>VM3</td><td>256</td><td>400</td><td>Web Server</td><td>180</td><td>797.61</td><td>1035.17</td></tr>
<tr><td colspan="7">Experiment1: Mixed Load 1 VM CPU Load,1 With Web Server running</td></tr>
<tr><td>Domain0</td><td>256</td><td>400</td><td>-</td><td>-</td><td colspan="2">NA</td></tr>
<tr><td>VM2</td><td>256</td><td>400</td><td>CPU</td><td>100</td><td colspan="2">NA</td></tr>
<tr><td>VM3</td><td>256</td><td>400</td><td>Web Server</td><td>180</td><td></td><td></td></tr>
</table>

The experiment as described in above table was done to analyze the effect of mixed load applications on the performance of each other. One application is a CPU intensive application and the other application is a web server. We carried out first experiment only with web server running in virtual machine vm3. Next experiment was carried out with CPU intensive application running in vm2 and vm3 is hosting the web server. In both the experiments we have not set the value of cap for the virtual machines. As shown in the following table, in both cases, CPU consumption by vm3 is the same which is 180% whereas in second experiment vm2 consumed 100% CPU. The test bed consisted of four cores of processor; hence there was still some CPU capacity left. But the readings show there is drastic change in throughput of the web server in the second experiment. Although CPU consumption is same in both experiments, the quality of service (QoS) delivered has gotten affected by the presence of the other virtual machine. The experiment described above was done with a simple setup. In a real life scenario, the situation can get worse in presence of tens or hundreds of virtual machines sharing the pool of resources. Each of the virtual machines may be hosting different kind of application with different kind of workload patterns and with different levels of desired quality of service. A change in any of the software components such as the virtual machine, or application characteristic or a change in any of the hardware resource can affect the performance adversely. Several studies [16] [17] [22] [27] [42] revealed that there is compelling need of having better performance isolation mechanism in Xen. This is also evident from the fact that three schedulers [47] named Borrowed Virtual Time (BVT), Simple Earliest Deadline First (SEDF) and Credit scheduler have been proposed for virtual machine scheduling in Xen in past four years. Lack of performance isolation causes degraded and unpredictable application performance. With this motivation, we define the problem in the following way.

D. *Problem Definition*

Our work is in the context of providing performance isolation across virtual machines sharing the resources. Specifically most important objective of our work is to devise a mechanism to set resource management parameters for the virtual machines in such a way that the applications running inside virtualized environment can deliver client QoS guarantees. The client QoS requirements need to be translated in resource management parameters. Another important objective is to improve resource utilization with constraint of maintaining client QoS. This objective is important from the perspective of the service providers. For example, the client QoS requirements can be expressed in terms of desired response time of the application. The resource management parameter to be tuned can be scheduler parameter cap of a virtual machine hosting the application. The value of cap represents the upper limit on CPU consumption by a virtual machine. The challenge is to design robust mechanism for setting up the cap of virtual machine in order to maintain the response time of the application even in presence of the other workloads or with the variations in the operating environment.

IV. PROPOSED METHOD

In this section we present our mechanism to compute the resource management parameters of the virtual machines so as to deliver QoS to the applications running inside virtualized environment. We applied the feedback control theoretic approach [35] for developing the solution. The basic idea of feedback control systems is that they work on the basis of the feedback they receive from the system at runtime. Therefore building a very accurate model of the system is not necessary. Also, as it works on feedback from a running system, it can respond quickly to the variations occurring in the system. Other alternative for developing the solution include queuing theory. But the queuing model does not handle feedback and it is not good at characterizing transient behavior in overload. Also a queuing model does off-line predictive analysis, whereas feedback control theory does online analysis which makes.

A. Feedback Control Theoretic Approach

As famous mathematician GEP Box said, all models are wrong, but some models are useful. As suggested by this quote [8], a mathematical model of a system may not be completely correct, but often the model is adequate enough to solve the specific problem. In control theoretic approach, we build the system models which approximately represent the effect of input parameters on the output metrics of the system. Using the system model, a feedback control system is designed. The online feedback from the system is monitored by feedback control system and accordingly the appropriate action to be taken is decided. The designed feedback control system can quickly react to any changes in the target system or in the environment by virtue of feedback supplied. Hence feedback control can be a good approach in the scenarios where a system is having several sources of dynamics. Let us go through the basics of feedback control theory to understand the solution approach in detail.

1. Elements of Feedback Control System

This subsection presents the working of a feedback control system. Figure 3 shows a basic feedback control system. A control system diagram is very different from a architectural diagram of a system. Control diagrams depict flow of the data and control signals through the system and the various transformations the signal undergoes. Architectural diagrams depict the functional components involved in the system. Some of the keywords used in feedback control theory are as follows:

- ➢ Target system: the system which is being controlled.
- ➢ Reference input: the desired value of the output metric from the system. This input may not be present in some scenarios. The subsequent part of this chapter will discuss that scenario in detail.
- ➢ Control error: difference between the values of reference input and measured output.
- ➢ Control input: variable whose value affects the behavior of the target system.
- ➢ Controller: controller is the most important component of a feedback control system. It computes the value of control input so as to maintain the measured output equal to reference input.
- ➢ Disturbance input: other factors that may affect the target system e.g. administrative tasks running on the same system as of target application under work.
- ➢ Noise input: noise represents an effect that changes the value of measured output produced by the target system.
- ➢ Transducer: Transforms measured output in some desired form. Transducer may be used for averaging of the output depending upon design of the feedback control system.

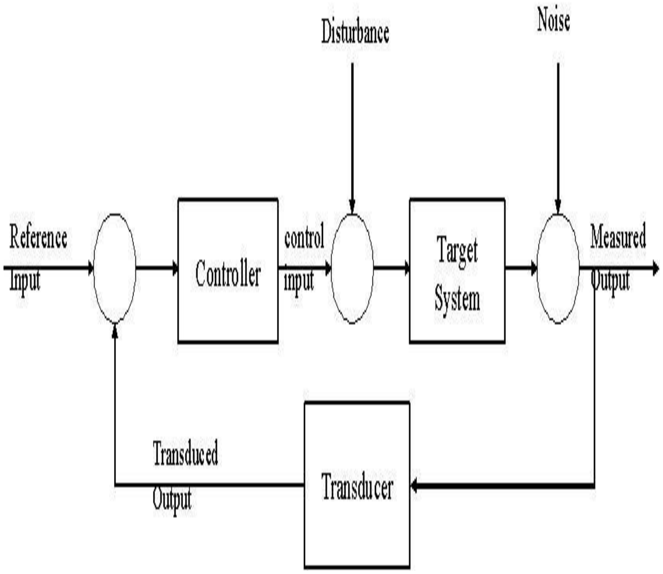


Figure 3. Typical feedback control system

The purpose of a controller which is called as control objective can be of following types.

- Regulatory control
- Disturbance rejection
- Optimization

Let us see how the control systems are developed with keeping these objectives into consideration. The control input parameters are the system variables or the configuration parameters which affects the working of the system which results in variations in the values of output from the system. The main idea in feedback control system is to monitor the output from the system and compute the new value of input parameters depending upon value of the current output. Task of controller is to model the input-output relationship for the system so that the desired responses from system can be achieved by setting up the proper values of input parameters.

B. Architecture of QoS aware Environment

Architecture proposed in our work is independent of virtual machine monitor (VMM) used, so we can use any Of the VMM solutions like VMware workstation, Xen, MS Virtual server. Figure 4.2 shows the architecture of QOS (quality of Service) Aware virtualized environment. Data centers host number of physical servers which are shared among multiple Client applications.

As shown in Figure 4 all the virtual machines consisting of tier1 of the application are placed on the physical server1, virtual machines of Tier2 on the physical server 2 and so on. Hence for n tier applications there will be at least n physical servers. Placement of these tiers is subject to resource availability on the given physical server. A virtual machine monitor will be running on each of the physical servers which do management of virtual machines on the given server. For simplicity we haven't shown the host OS or VMM in the given architecture. Please refer to figure 4.2 for the Architecture of the virtualized environment with VMM and host OS. Apart from these usual components of the virtualized environment, we add three modules named controller, capacity Analyzer and sensors. Sensor module is deployed in the tier 1 of all applications. As the name suggests, the Task of the sensor is to carry out measurements. Sensor will monitor each request coming to the application and Measure the values of interest. The measured values can include QOS parameters like response time delivered to each request, throughput of the application. The other task of the sensor will include transforming the measured Output in some form which is further being used by controller. The transformation can include summarizing the measured data, storing the history data etc. The controller and capacity analyzer modules are deployed in the host OS on each of the physical server. Controller module receives the values of the QOS parameters from the sensors. Task of the controller is to compute the new values of the resource management parameters for the virtual machine. In this architecture, we compute the Resource management parameter values for each virtual machine separately. The computed values for each of the Virtual machine are then supplied to capacity analyzer. Capacity analyzer verify whether the resource demands of All virtual machines together will get satisfied on the given physical server or not. Note that each physical server will have separate instances of controller and capacity analyzer running. After verification from the capacity analyzer, The resource management parameter values are then forwarded to the virtual machine monitor which acts as actuator to set these values. Following subsection describes the feedback control system covering these three Modules in depth.

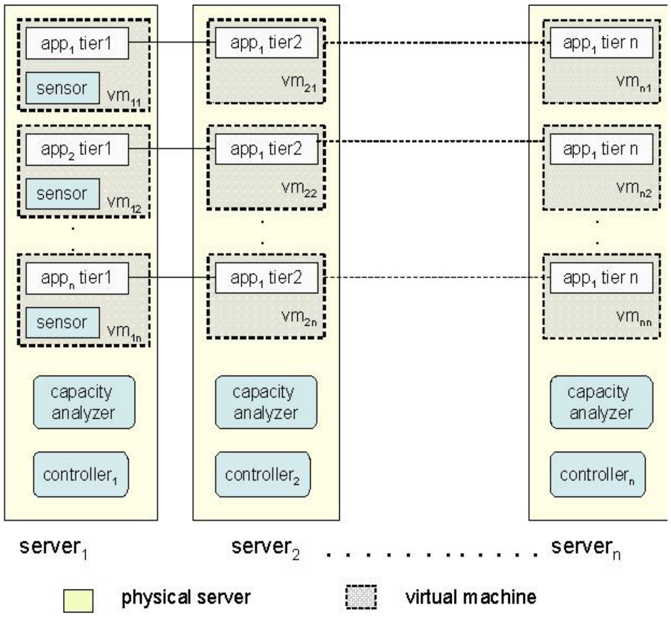


Figure 4. Architetcure of QoS aware environment

C. Feedback Control System

The following figure 5 depicts the design of the feedback control system for virtualized environment. For simplicity we are assuming number of applications and number of tiers of every application to be 2 each. Note that each physical server will have separate instance of this feedback control system. For this study we focus on maintaining the response time delivered by application. Response time is the measurement of time between arrival of the request at the server and departure of the request after successful service from the server. Delay over the network between the server and the client is not included in the response time measurement. Hence we are having one reference input in the form of desired response time for an application. In this study we are using cap of the virtual machine hosting the application as control input. Cap of the virtual machine puts the upper limit on the CPU consumption by a virtual machine. We are modelling the system using multiple SISOs. SISO stands for single input single output system. There will be one SISO for one virtual machine of each application running on a physical server.

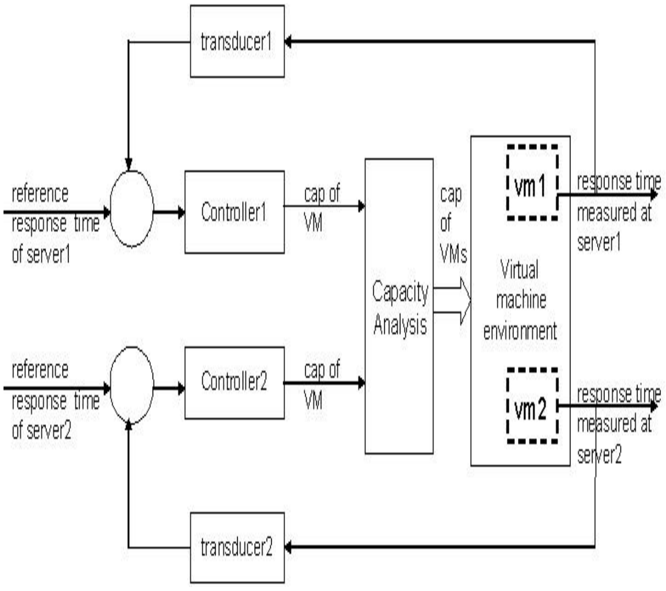

transducer1
reference response time of server1
Controller1
cap of VM
Capacity Analysis
cap of VMs
vm1
Virtual machine environment
response time measured at server1
reference response time of server2
Controller2
cap of VM
vm2
response time measured at server2
transducer2

Figure 5. Feedback control system for virtualized environment.

As shown in the figure, virtual machine environment is hosting two applications in different virtual machines. Feedback control system gets desired response time for each of the application as the reference input from the user. This input is entirely choice of the user which describes desired Quality of Service. Response time delivered by each of the application is measured with sensors present in the virtual machines. This measured output is then given to transducer which computes exponential average of the response time. Exponential averaging is useful in order to avoid responding to the temporary fluctuations in the system. Exponential averaging technique updates the average response time value in following manner:

Avg_response_time = α * current_response_time + (1 - α) * old_avg_response_time.
where α denoted exponential factor. Value of α can be configured by the system administrator depending desired responsiveness to the changes in the system.

The exponentially averaged response time value is provided to the controller along with the desired response time value. We implemented a PID (Proportional-Integral-Derivative) controller. The controller computes the new value of cap for the virtual machine. The controller computes the cap for the two applications separately. Hence logically there are two controllers running on a given physical server, so we have shown two controllers in this figure. The values computed by both controllers is feed to the capacity analyzer which verifies whether the resource demands of the virtual machines running on same physical server are feasible or not. If the resource demands exceed the capacity of the physical server then we need allocate some more hardware resources or we should discard some workloads. Allocating new hardware resources can be done by migrating the virtual machines on different physical server. The virtual machine migration technology is supported by many of the virtual machine monitors. Virtual machine migration allows runtime migration of a virtual machine from one physical server to other physical server.

D. Optimal Control Design using Linear Quadratic Regulator

According to feedback control system there is pre-specified QoS requirement for each application. However, in many situations these requirements may not be explicitly specified by the client of the data center. Rather the explicit requirement can be to maximize the application performance at minimal cost. The cost in the context of these software applications is the amount of resources used by the application. The service providers charge the clients according to usage of resources by the client application. Hence goal here is to maximize the application performance with minimal usage of resources. Here we are focusing on cpu sharing among the virtual machines. Hence the goal of this work is to maximize the application performance with minimal cpu usage by an application.

1. Optimal Resource Allocation

Our goal is to develop a mechanism to find optimal resource allocations for the virtual machines hosting the applications. Let us go through the scenario of virtual machines in detail. As we discussed earlier, the credit cpu scheduler of Xen accepts two parameters per virtual machine: the weight and the cap. The cap of a virtual machine is the upper limit on cpu consumption by the virtual machine. Hence an increase in the cap of a virtual machine may improve the performance of the application running inside the virtual machine. We need to compute the value of cap where the application performance is maximized with respect to resource usage. The ratio of client interest can be expressed in following form:

$$\frac{(\boldsymbol{c1*throughput})}{(\boldsymbol{c2*response_{time}})+(\boldsymbol{c3*allocated-cpu-capacity})} \qquad (1)$$

Our aim is to compute the value of cap of virtual machine for which this Ratio is maximized. Note that the variables c1, c2, c3 represents the relative weight of each metric. Hence for every set of values of relative weights we may have different values Of cap at which the this ratio is maximized. The application performance metrics considered in this work are response time and throughput of the application. The objective of this work is to dynamically compute the value of Cap of a virtual machine hosting an application such that:

- The response time of the application is minimized.
- The throughput of the application is maximized.
- The value of cap allocated to the virtual machine is minimized.

Hence we need to design a control system which will continuously monitor the virtualized Environment and set the cap of a virtual machine in such a way that the application. Performance is always maximized. This control system needs to be capable of discarding the effect of disturbance tasks running on the same system. We use the LQR (Linear Quadratic Regulator) to develop the solution for this problem.

2. Linear Quadratic Regulator

The LQR (Linear Quadratic Regulator) [28] [16] is suitable for solving optimization problems using control theoretic approach. The name LQR comes from the fact that the dynamics of the system can be represented by linear difference equations and the goal for LQR controller is to minimize a quadratic cost function. The cost function is generally expressed in terms of control error and control effort. The LQR controller computes the values of control input in such a way that the cost function is minimized. Let us go through an example to understand the details of LQR.

Consider a MIMO (multiple input multiple output) system which can be represented by a linear difference equation such as: $x(k + 1) = Ax(k) + Bu(k)$. Here x(k) represents the value of state space vector of the system at time instant k and u(k) represents the value of the control input vector at time instant k. A,B are the system model parameters. The cost function to minimize for this system is given as follows:

$$J = \frac{1}{2}\sum_{0}^{\infty}[x^T(k) * Q * x(k) + u^T(k) * R * u(k)] \qquad (2)$$

Here Q, R are the matrices representing the relative cost of control error and control effort respectively. Q must be a positive semi definite matrix and R needs to be a positive definite matrix. This condition ensures that the value of J will be non-negative. R is relative weight of the control effort which should not be neglected in the cost function, hence R matrix need to be positive definite. The LQR algorithm computes the value of controller gain using the values of matrices Q,R and system model parameters A,B. However, LQR needs the input of weighting matrices Q and R values which ultimately affects the controller parameters. In this work, we used the some simple techniques to set the values of Q and R matrices.

3. *Feedback Control Design for Optimal Control*

Let us describe the proposed method for providing optimal control to the applications running inside the virtual machines. The solution is proposed for a single application hosted in a virtualized environment. Figure 4.1 shows the testbed setup for the proposed solution. We deployed the two-tier Web application using two virtual machines located on two different physical servers. The Apache server is running on virtual machine VM1 which is hosted on physical Server 1. MySQL server is running on virtual machine VM2 which is hosted on physical Server 2. As described in earlier chapter, this setup also contains the sensor in the form of a Muffin proxy server running on VM1. The controller is running on physical Server 1. The utilization of database server is very low, hence we have not considered tuning of cap of corresponding VM2. Random request URLs are generated using a client machine running httperf load generator.

The performance metrics considered in this solution are response time and throughput of the application. Hence the state space of the system can be given by:

$$x = \begin{bmatrix} response\ time \\ \dfrac{1}{throughput} \end{bmatrix}$$

As there is only one virtual machine whose cap needs to be tuned, the control input vector is given by:

$$u = \begin{bmatrix} Cap \\ vm1 \end{bmatrix}$$

The system model is given by the equation: X(k + 1) = AX(k) + Bu(k). The model parameters A,B can be computed using a system identification process.

The proposed feedback control system for optimal control is shown in the following diagram.

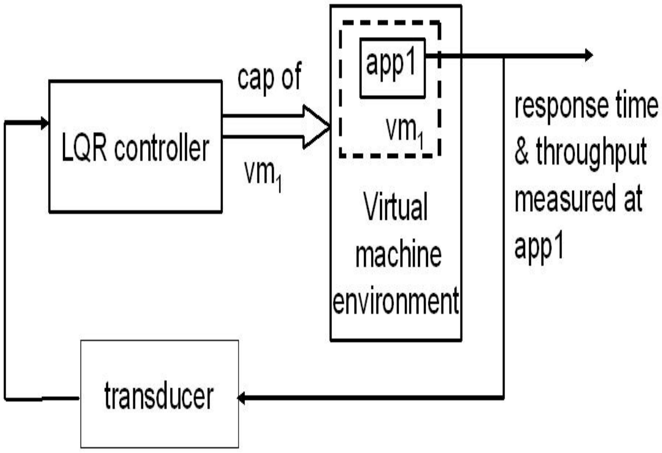

cap of
vm1
LQR controller
app1
vm1
Virtual machine environment
response time & throughput measured at app1
transducer

Figure 6 Architecture of Optimal Control using LQR

As shown in the Figure6.1, there is no reference input to the system. The goal here is to set the value of cap such that the cost function given in Equation 4.2 is minimized. Hence the measured output here is the response time and throughput of the application. These values are provided to the transducer which in turn computes the exponential average of both the metrics using the formula given by Equation 3.2 in Chapter 4. These averaged values are then provided to LQR controller which computes the value of cap to set in order to minimize the cost function for the system which can be given as follows:

$$J = \frac{1}{2} \sum_{0}^{\infty} \left[\left[response\ T(k) \quad \frac{1}{throug\ hput\ (k)} \right] * Q * \right.$$

$response \quad T(k)1throughput(k)+ \quad Cap \quad vm1$
$k*R*Capvm1(k)$ (3)

Where response T(k) represents the value of response time at time instant k and throughput(k) is the value of throughput of the application at time instant k. Note that instead of control error, the -state space of the system represents the actual values of output of the system. Also the control effort capvm1 is the absolute value of the control input and not the deviation from some steady state value. Q needs to be 2 × 2 matrix and R needs to be 1×1 matrix. We can see that this Equation 6.3 gives the effect similar to maximizing our example ratio 6.1.

Ratio=

$$\frac{(\boldsymbol{c1*throughput})}{(\boldsymbol{c2*response_{time}})+(\boldsymbol{c3*allocated-cpu-capacity})}$$

The control law used in this controller is given by:

$$u(k) = -K * X(k)$$

The value of feedback gain matrix K is computed using the LQR algorithm. The algorithm takes the values of system model parameters A,B and weighting factors of cost function Q,R matrices as the input. The MATLAB [23] command dlqr implements the LQR algorithm. We used this command to get the value of K matrix. Hence for developing the feedback control system for optimal control following steps need to be followed:

1. Perform the system identification experiments to construct the system model. This step gives us the A and B matrices.
2. Assign the values to weighting matrices Q,R according to desired tradeoff between CPU usage and the application performance to be delivered.
3. Compute the value of feedback gain matrix K using LQR algorithm. The first three steps can be done offline whereas the fourth step is online.
4. Implement the LQR controller with the value of K matrix computed in previous step.

Hence with these steps, the LQR controller can be deployed in the host OS of physical Server 1.

V. EXPERIMENTAL SETUP

This section describes the Experimental setup (testbed deployed) for carrying out the experiments. We designed and deployed components in the testbed in a way so as to resemble to real world scenario. For building the testbed, we have used open source solution Xen3.0.3.

A. Components of Testbed

For demonstration of the work we have used two-tier systems with apache web server at frontend and MySQL database server connected at the backend. Apache server hosted the two-tier Web application which has web and database tiers. We used httperf for load generation. We have used two instances of the same two-tier system to demonstrate how we can deliver differential quality of service to each of the application. We created four virtual machines by using Xen. Two of the virtual machines are hosting one apache server each and two other virtual machines are hosting one MySQL server each. Fig 5.1 explains the Testbed for QoS aware virtualized environment.

Following describes the hardware components of the testbed and how the software components are deployed on the hardware. The testbed setup is shown in the figure 7. Our testbed consists of two machines each with following configurations are used for hosting the servers.

- Server1: Intel(R) Xeon(TM) dual CPU 2.80GHz processor, 2 GB main memory.
- Server2: AMD Athlon(tm) dual core processor 3.0GHz, 1 GB of main memory.

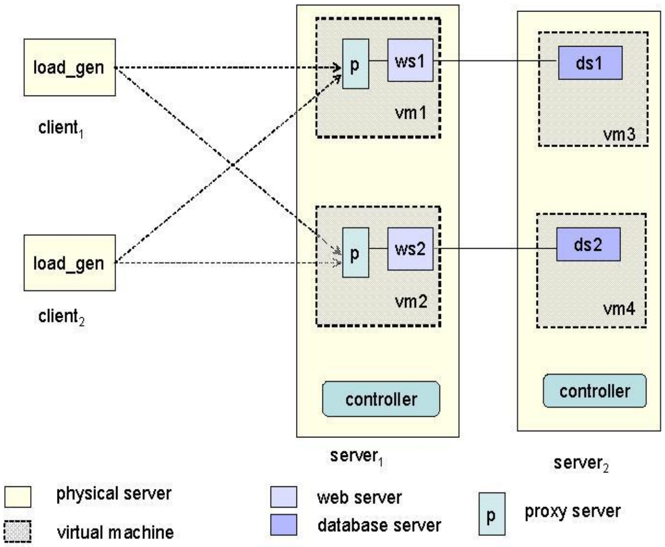


Figure 7. Testbed for QoS aware virtualized environment.

Generally data centers put same tiers of different applications on the same physical server. We adopted this design by putting virtual machines hosting the web tiers on server1 and virtual machines hosting the database tiers on server2. Apart from the above data center design, we have used 2 client machines to emulate behavior of real workload with the help of continuous load generation using httperf [47]. Requests are having exponential distribution. All of the machines are running with linux2.6. All of the machines are connected with 100Mbps Ethernet. we designed two controllers each of which is running in the host OS on each of the physical servers. Each of the virtual machine hosting the web tier also hosts a http proxy named Muffin which acts as sensor. Muffin simply forwards the requests coming from the clients to the web server. We have modified the source code of Muffin to measure the response time of the web server. This proxy acting as sensor gives the response time measurement to the controller running in the host OS. This controller also communicates these response time values with other controller running in the host OS on server2 hosting the virtual machines corresponding to the database servers. The proxy Muffin is written in java, whereas all the utilities required for extracting the response time values from muffin log files, controller design is done by coding in C and shells script. Communication among the machines for exchange of the values and parameters is done using sockets programming. For deploying Web application, we installed apache web server, php on the virtual machines hosting the web tier. Also we installed MySQL on the virtual machines hosting the database tiers.

Following diagram shows flow of a request coming to a application1 running inside our testbed. As shown in last figure of testbed, application1 has its web tier running inside the vm1 and database tier running inside vm3. The virtual machines vm1 and vm3 are running on two different physical servers.

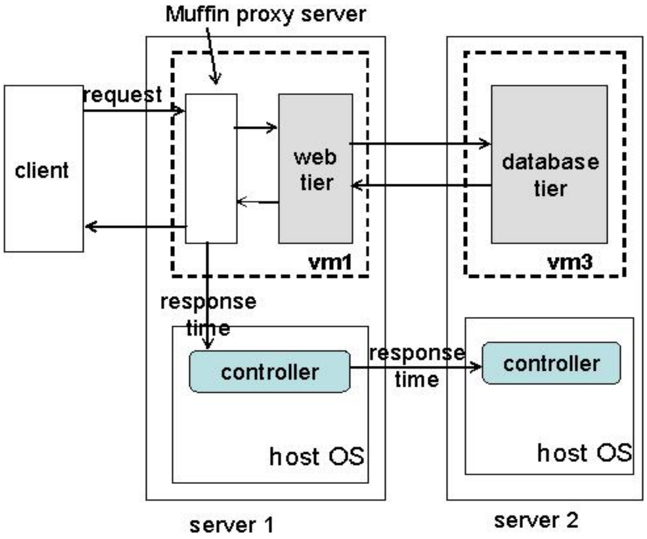


Figure 8. Response time measurement and flow of a request through the testbed

B. Workload Description

The nature of the workload deployed in the virtual Machines has an impact on the behavior of the QoS delivered. The resource usage pattern of one VM affects the performance of application running in other VMs. Hence we deployed Web application which is two tier applications. We deployed the two tiers in two separate virtual machines which are hosted on two different physical machines which depicts the practical scenario in the data-centers. This workload exercises different IO tasks like querying database, flow of requests through network as two tiers of a application are located in two different virtual machines.

VI. PERFORMANCE EVALUATION

A. Execution Architecture of Optimal Control System using LQR

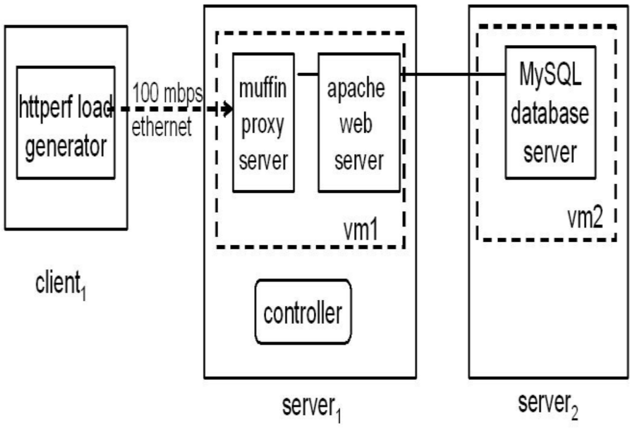


Figure 9. Optimal Control System design Architecture

This section describes the he Execution Architecture (testbed deployed) for carrying out the experiments. The following diagram explains the architecture of Optimal Control System using Linear Quadratic Regulator (LQR).

The Figure 9 explains Optimal Control System architecture.

We carried out the system identification experiments on the testbed described in Figure 11. The physical Server 1 has two core CPU, hence the maximum cap of a virtual machine can be set to 200. Here in the1se experiments, the cap of virtual machine VM1 is varied between 10 to 190%. Each value of cap is set for a period 20 cycles of 10 second duration each. During this period, the values of response time and throughput are monitored and recorded to the log files.

The plot in Figure 10 shows the results of this experiment. Some observations can be made from this plot:

- The response time and throughput curves are linear in the cap range of 10-110%. The response time values decrease linearly whereas the throughput values increase linearly in this region.
- The response time and throughput values remain almost constant during the cap range of 120-180%.
- The response time and throughput degrade rapidly when the cap value is below 50%.

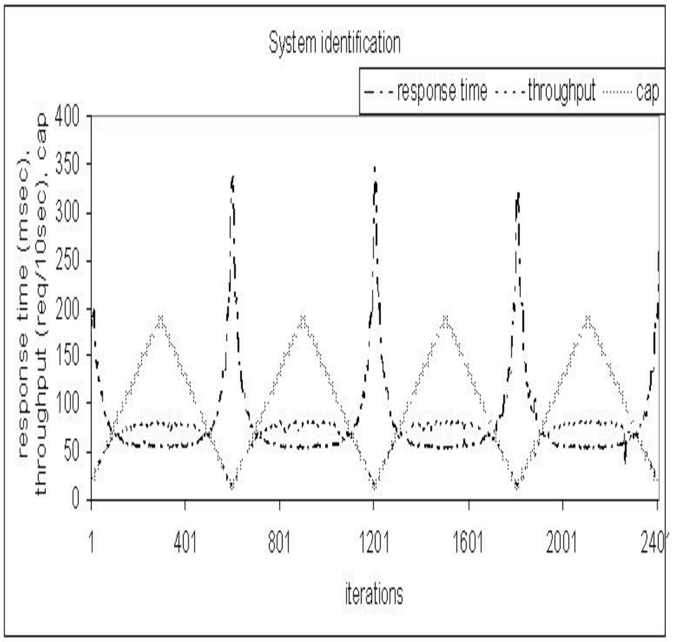


Fig 10 System identification experiment results on Test bed for self tuning optimal control

Using the measurements recorded in this experiment we computed the values of system model parameters A,B which are used in the system model

X(k+1) = A*X(k)+B*u(k).

The values obtained are: A = [0.9910 0.0501; 0.0987 0.5551] and B = [−0.0052; 0.0048]. Subsequently we need to assign the values to the weighting matrices Q, R so that controller parameters can be computed.

B. Performance Analysis of Optimal Control using LQR

The last step for designing the LQR controller involves fixing the values of weighting matrices Q,R. This is the most important step and it directly affects the controller behavior. The values of Q,R represents the trade-off between the application performance and the cpu usage. We choose the matrix q as [2 0; 0 50] and matrix r as 2.

The following diagram explains the cost function for optimal control in virtualized environment.

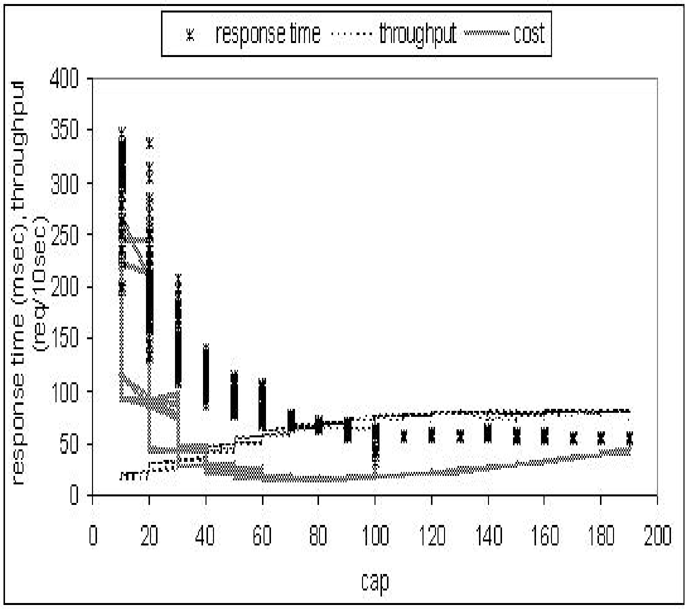


Figure 11. Cost function for self tuning optimal control in virtualized environment-setting1

Let us call this set of values of Q and R as setting-1. The plot in figure 7.3 shows the cost function curve against the cap of virtual machine. For simplicity while plotting and analyzing the graphs, all the cost function curves are scaled down in the plots. The values plotted in this graph are instantaneous values of the cost function and not the value of actual cost function j which is given by equation 6.2 which involves summation of all terms. The cost function, the response time curve and the throughput are plotted on y-axis against the values of cap of virtual machine vm1 on x-axis in increasing order. It can be observed from this graph, that the value of response time is very high for the initial part. Also the throughput is very low in the same region. Hence the value of cost function is also high in that region. As the application performance is not good enough, the cost function has higher value in this region. As the response time starts decreasing and throughput starts increasing, the value of cost function also goes down. In the region where the cap value is between 70 to 110% the cost function has the minimum cost. The cost function curve rises after this region. This rise is due to the fact that increasing the cap of virtual machine does not result in significant improvement in performance. The cost function curve correctly depicts the cost factor from client perspective. Hence the region of cap around 70 to 110% is the one with the minimum cost for the client application. This is the optimal region which satisfies the desired properties we stated in the problem statement. The feedback gain matrix K is computed using the system model parameters and a few initial iterations the response time and throughput value settles down. Irrespective of starting value of cap, controller sets the cap to its optimal value in a few iterations. In this experiment the initial value of cap was 80. Weighting matrices are Q, R. The computed gain matrix k is [-1.9920 -0.1423]. We implemented the LQR controller with this value of feedback control gain and performed the experiments to evaluate the controller. The graph in figure 12 shows the values of response time and throughput delivered by the application.

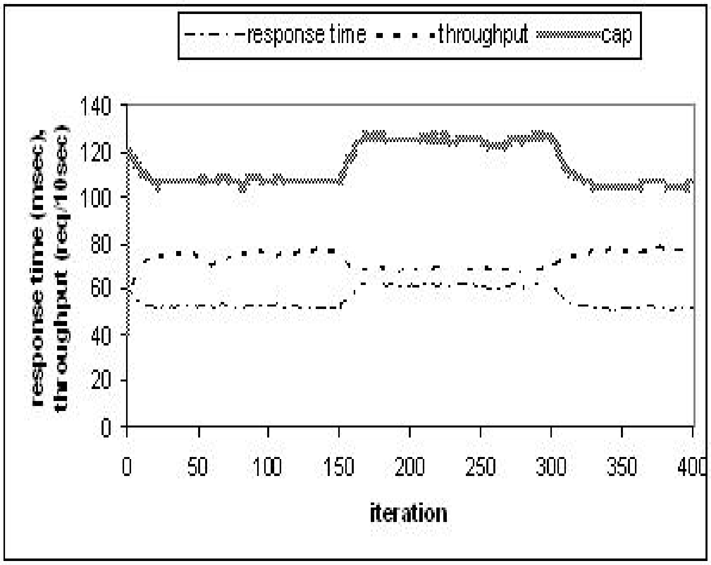


Figure 12. Performance evaluation of self tuning optimal control in virtualized environment (response time throughput delivered by application)-setting1

The graph also shows the plot of cap of vm1. The iterations are plotted on the x-axis whereas the application performance metrics and the cap is plotted on y-axis. It can be observed from the graph that after a few initial iterations the response time and throughput value settles down. Irrespective of starting value of cap, controller sets the cap to its optimal value in a few iterations. In this experiment the initial value of cap was 80. We may verify whether the state of the system is optimal by comparing the controller results from figure 12 with cost function curve in figure 11. As we can see that the cost function has the minimal value in the region of cap values between 70 to 110%. The optimal value of the cap set using the LQR controller is 103-105% which is under the minimal cost range. It can also be observed that the response time values delivered are around 52-55msec which is the same range of values we observed in the optimal region in the figure 7.4. Throughput is also observed to be around 65-67 req/10sec which is the same range of values delivered in the optimal region in the figure 7.3. This verifies that LQR controller is able to drive the system near to the optimal state. We introduced a disturbance task on the virtual machine vm1 where the application is running. The disturbance task is a cpu-intensive task which alternately executes some computation and sleeps. As we can see from the graph, the disturbance task was introduced around the 150th iteration. Due to this disturbance task, the value of response time increased and the throughput decreased. The controller reacts to this change in the system and the value of cap is increased due to increased cpu demand. However, due to the presence of disturbance the application performance does not remain same as its previous value. The system enters into a new state where the values of cap and application performance are different from the ones before the disturbance. As we can see the graph in figure 12, the disturbance task is removed after 300th iteration and the cap value is restored to its previous optimal value. The application performance metrics also attain the previous optimal values.

C. Sensitivity Study of Optimal Control

The controller behavior depends on number of factors such as the values of weighting factor, incoming load on the application. A change in any such factor may cause the change in value of optimal cap set by the controller. Here we carry out experiments with different settings to find the effect of change in these factors. We also compare the controller results with the cost function curves for each setting to verify whether the controller has been able to keep the system near optimal region or not. In this work, we considered weighting factors and load levels as the parameters to carry out the sensitivity analysis.

1. Sensitivity of optimal controller to weighting factors of cost function

As we have discussed in earlier sections, the value of cost function and ultimately the controller parameters are sensitive to the values of Q, R matrices. To gain some insight into the cost function value, we have plotted values of individual terms in the cost function. There are three terms involved in cost function given in Equation 4.2, which are the response time, throughput and cap respectively. For simplicity, we kept values of elements Q12 and Q21 of Q matrix to 0. Following are the three terms:

- Response time term: Q11 * response time2
- Throughput term: $Q22 * (\frac{1}{throug\ hput})^2$
- Cap term: R *cap2

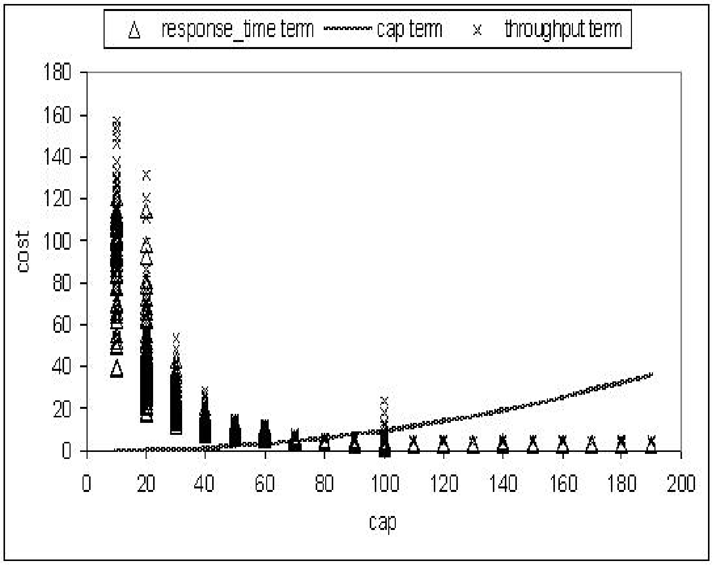


Figure 13. Decomposition of cost function for self tuning optimal control in virtualized environment- Setting1

The graph in Figure 13 shows the plots of these three terms. The response time term, the throughput term and the cap term are plotted on Y-axis against the cap on the X-axis. It can be observed from the graph that in the cap range of 10-50% the significant part of cost comes from the response time and throughput terms. In the cap region of 60-100% all the three terms equally contribute to the value of cost function. For the cap values above 110%, the cap term is major part of the cost function. These plots of individual terms have these specific curves due to the values of Q,R matrices. These plots and as well as the total cost function plot show sensitivity to changes in Q,R matrices. Let us go through some different sets of Q, R matrices and their effect on the cost function curve and the overall controller behavior. We will now see the effect of change in the value of cap term by changing value of R from 2 in setting-1 to 10 in setting-2. For observing the effect of change in response term, we change the value of Q11 from 2 in setting-1 to 0.5 in setting-3. Table 2 lists out these three settings. Let us now go through each of these settings.

Experiment settings for studying the sensitivity of controller to weighting factors described in table2

Setting No	Q Matrix	R	K	Expected Optimal Cap region
1	**[2 0; 0 50]**	**2**	**[-1.9920 -0.1423]**	**70-110%**
2	**[2 0; 0 50]**	**10**	**[-1.2379 -0.1224]**	**50-70%**
3	**[0.5 0; 0 50]**	**2**	**[-1.7416 -0.1145]**	**60-90%**

In the setting-2, we changed the values of Q and R matrices as Q = [2 0; 0 50] and R = 10. The new value of feedback control gain K matrix is [−1.2379 − 0.1224]. The graph in Figure 7.7 shows the cost function curve, along with plots of response time and throughput values against the cap values plotted on the X-axis. It can be observed from the graph that the cost function has minimal values around the region of cap values 50-70%. We evaluated the system performance with the new controller parameters. The graph in Figure 15 shows the plots of application performance metrics and the value of cap set by the controller against the iterations on the X-axis. It can be observed from the graph that the cap value is settled around the value of 82% which is slightly higher than the expected minimal cost region of 50-70%.

In setting-3 we changed the values of Q, R matrices with Q = [0.5 0; 0 50] and R = 2. The new value of feedback control gain K matrix is [−1.7416 − 0.1145]. The graph in Figure 7.9 shows the cost function curve, along with plots of response time and throughput values against the cap values plotted on the X-axis. It can be observed from the graph that the cost function has the minimal values in the cap region of 60-90%. The graph in Figure 7.10 shows the plots of application performance metrics and the values of cap set by the controller against the iterations on X-axis. It can be observed from the graph that the cap value is settled around the value of 100% which is slightly higher than the expected minimal value region of 60-90%.

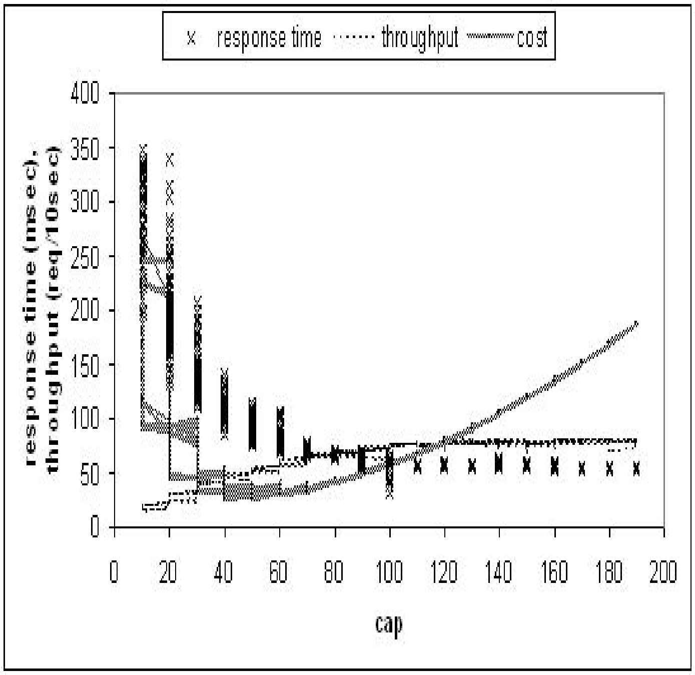


Figure 14. Cost function for self tuning optimal control in virtualized environment - setting-2

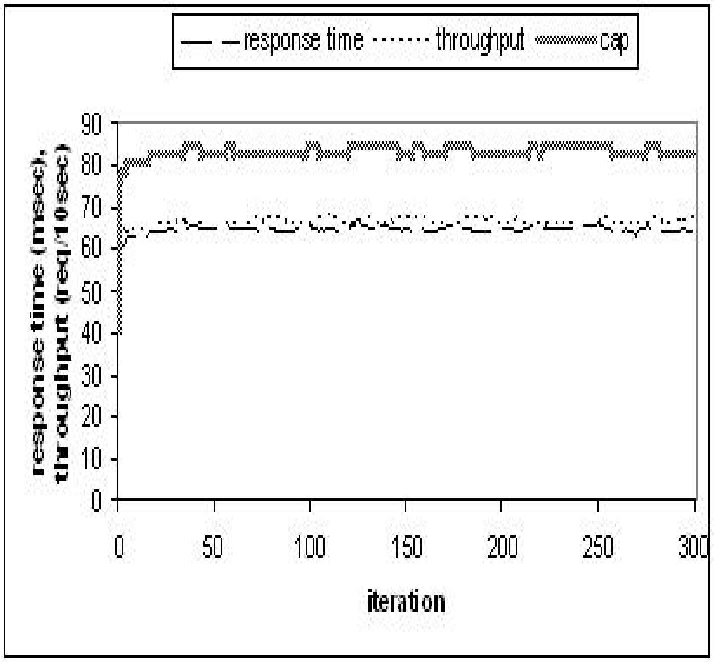


Figure 15. Performance evaluation of Self tuning Optimal control in virtualized environment : setting-2

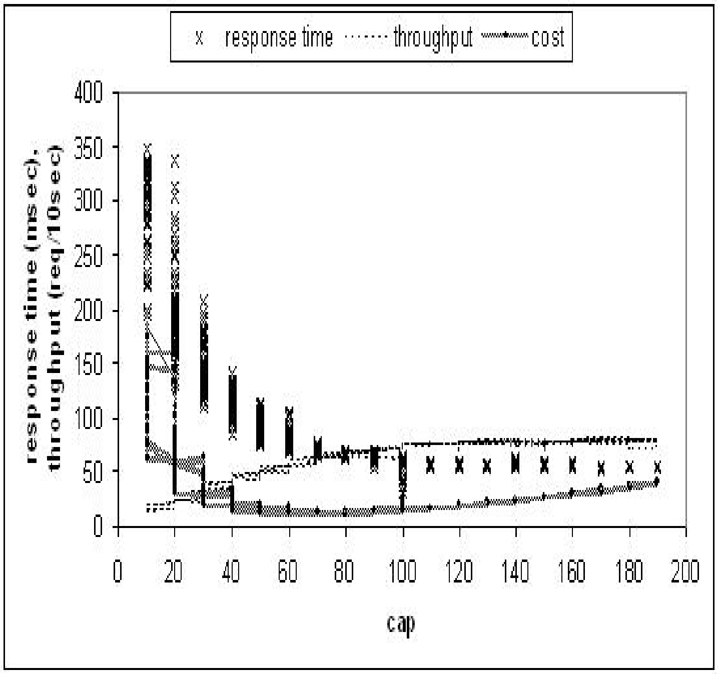


Figure 16. Cost function for self tuning optimal control in virtualized environment: setting-3

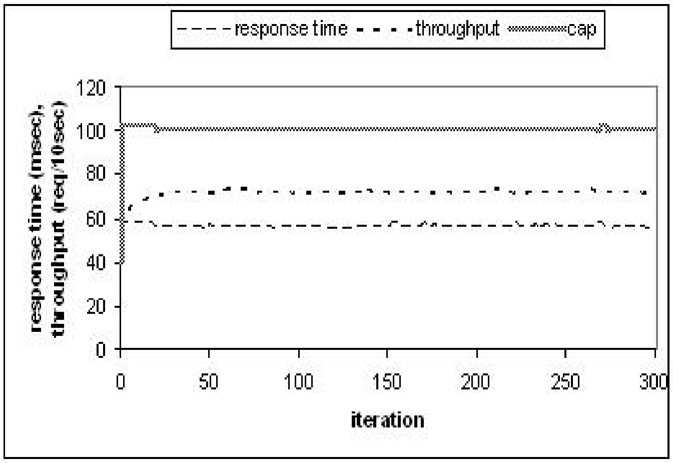


Figure 17. Performance evaluation of self tuning optimal control in virtualized environment: setting-3

The following Table 3 shows the summary of the different settings of Q and R matrices and the sensitivity of application performance to these settings. Let us discuss these different settings and their effect on controller behavior.

Performance evaluation of optimal controller with different settings of weighting factors explained in table3

Setting No	**Q Matrix**	**R**	**K**	**Expected Optimal Cap region**	**Optimal Cap Value Set in Experiments**
1	[2 0; 0 50]	2	[-1.9920 -0.1423]	70-110%	107-108%
2	[2 0; 0 50]	10	[-1.2379 -0.1224]	50-70%	72-74%
3	[0.5 0; 0 50]	2	[-1.7416 -0.1145]	60-90%	100%

In the setting-2, the value of R is increased to 10 from 2 in the setting-1. This caused the cap term to become more significant. Hence the cost function value started to rise just after the value of cap reached 70%. In setting-1, the cost function was less sensitive to the cap value compared to setting-2. Hence cost function had minimal value in higher cap region which is 70-110%. In setting-3, the value of parameter Q11 associated with the response time was decreased from 2 in setting-1 to 0.5. Hence now the cost function became lesser sensitive to the response time and the other two terms dominated over the response time term. This resulted in new optimal region where the response time is slightly poor than its value in setting-1.

The choice of values of weighting factors Q and R has an impact on controller behavior. As the weighting factors Q and R matrices are relative to each other, modifying the value of one of the elements has the effect on all other terms. To understand the effect of individual terms in these matrices it is important to perform the experiments with different settings of these matrices. The starting point for choosing the values for Q and R matrices can be based on two things -

- The possible range of values of the state space vector elements and of the control input.

- The desired trade-off between the CPU usage and the application performance. Among these two things, the desired trade-off needs to be set by the administrator of the system. However it can be difficult to quantify this trade off. Hence some experimentation is needed to come up with the quantified trade-off. Also the experimentation is needed to know the range of the state space elements.

2 Sensitivity of optimal controller to different levels of load

In this section, we will discuss the experiments carried out to study the effect of different levels of load on optimal controller behavior. We carried out the experiments with four different levels of load. The experiment setup is same here as described in previous section. We are using setting-1 of the Q and R matrices in which the values of these matrices are given by Q = [2 0; 0 50] and R = 2. The value of feedback gain matrix K is K = [−1.9920 − 0.1423]. Table 4.3 gives the summary of these experiments.

The first entry represents the experiment carried out in setting-1 in previous section. The rest of the entries belong to different load levels such as 85 req/10sec, 69 req/10sec and 25 req/10sec. We also carried out experiments without any controller running to find out optimal cap value for each load level. We applied the cost function to the measured values of each experiment and plotted the values. The optimal cap value obtained from these cost function curves is used to verify whether the controller is able to set cap to the optimal value corresponding to each load level.

Performance evaluation of controller with different levels of load explained in table4

Setting No	Average load (Throughput in req/sec)	Average Response time (msec)	Expected Optimal Cap	Average Cap Set By Controller
1	73.68	52.07	70-110%	106.47
2	85	55.15	70-110%	112.42
3	68.90	50.72	50-100%	103.96
4	25	46.55	70-100%	99.02

The setting-1 has already been covered in last section. We observed that the expected optimal cap value was 70-110% and the average cap value set by the controller was 106.47%. In case of setting-2 the load has been increased to 85 req/10sec. It can be observed from the graph in Figure 18 that the cost function has minimal value in the region of 70-110% cap. The graph in Figure 19 shows the cap values set by the controller when the load of 85 req/10sec is applied to the system. The average value of cap set by the controller is 112.42%. In setting-3, the load on the application is reduced to 65 req/10sec. The optimal cap value in this scenario is in the range of 50-100% whereas the average value of cap set by the controller is 103.96%. The graphs in Figures 20 and 21 shows the plots of the values observed with the setting-3. We reduced the load to 25 req/10sec in setting-4. The offline experiments show that the optimal cap value is in the range of 70-100%. The average value of cap set by the controller is 99.09%. The graphs in Figures 22 and 23 shows the plots for cost function and controller evaluation for setting-4. The cap values tuned by the controller in each of these setting are very close to the expected optimal values in the respective case. At each load level, the controller is able to drive to the system near to the optimal state. It can also be observed that at each load level, for the given optimal cap value the average response time for the application is also different. The average response time values are in the range of 55 to 46 msec. Hence from all these observations we can conclude that this feedback control system is able adjust the cap of the virtual machine near to its expected optimal value even with the changes in the load levels.

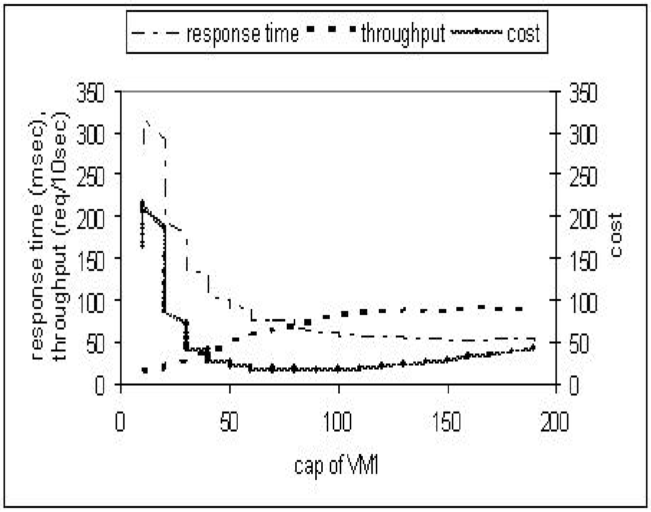


Figure 18. Cost function for optimal control in virtualized environment: setting-2

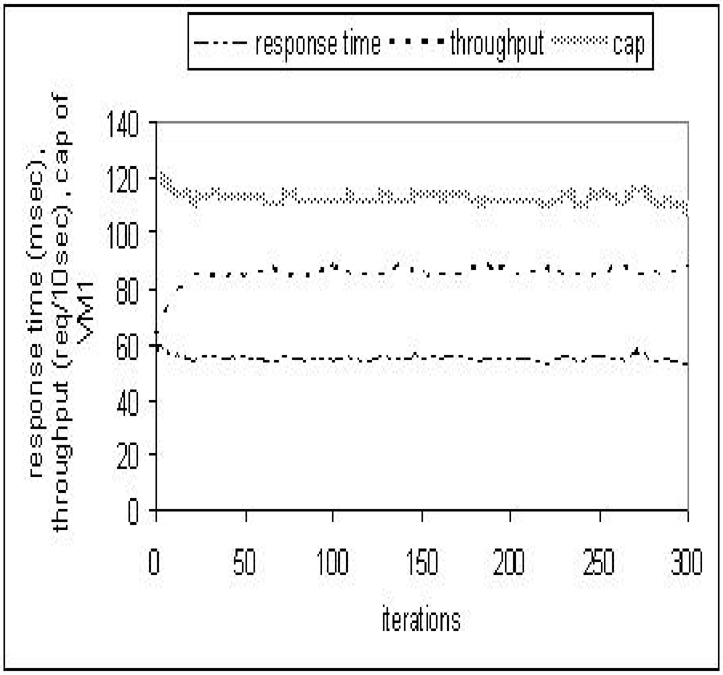


Figure 19. Performance evaluation of Optimal control in virtualized environment : setting-2

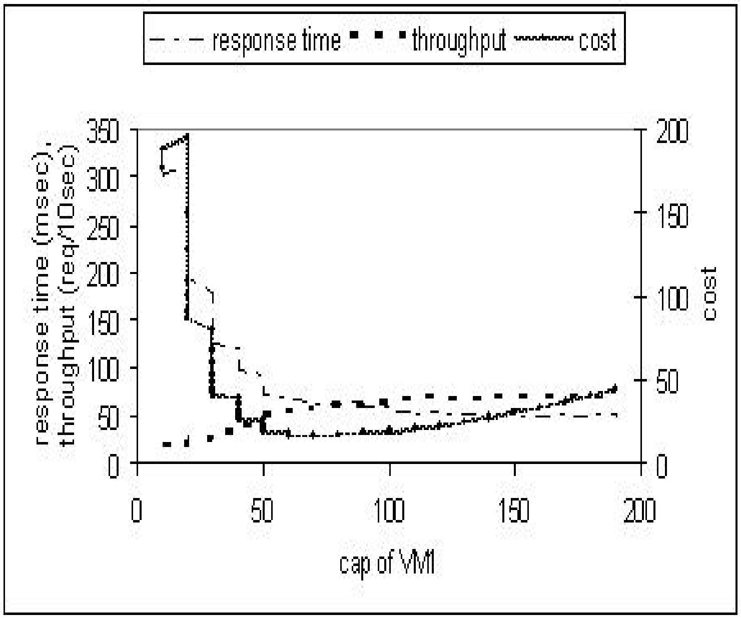


Figure 20. Cost function for optimal control in virtualized environment: setting-3

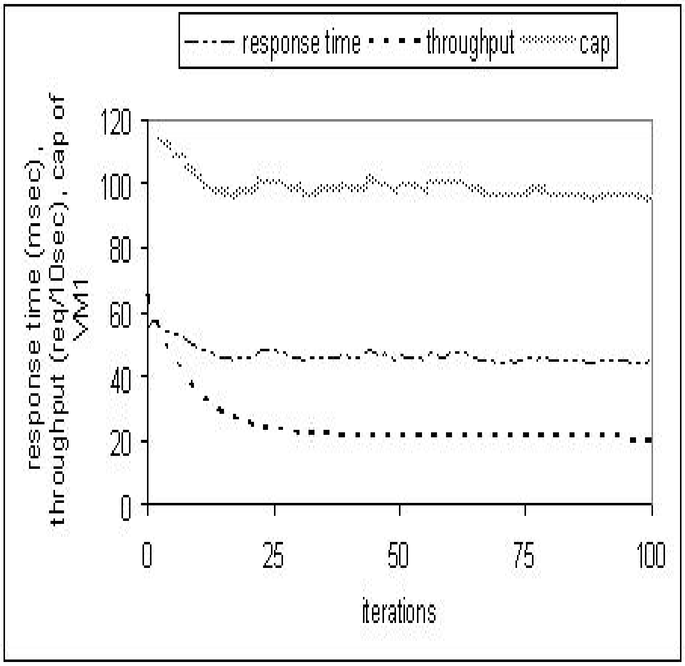


Figure 21 Performance evaluation of Optimal control in virtualized environment:
setting-3

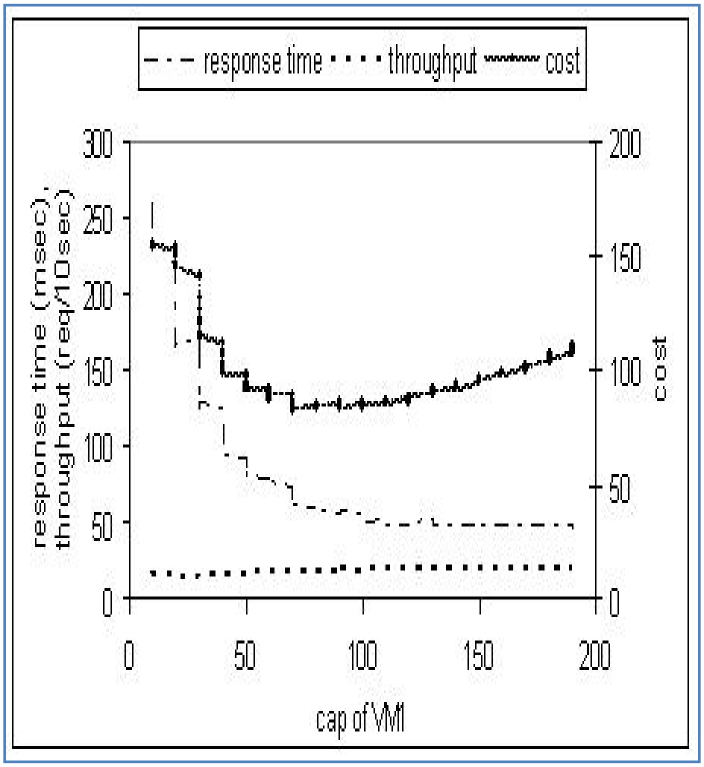


Figure 22. Cost function for optimal control in virtualized environment : setting-4

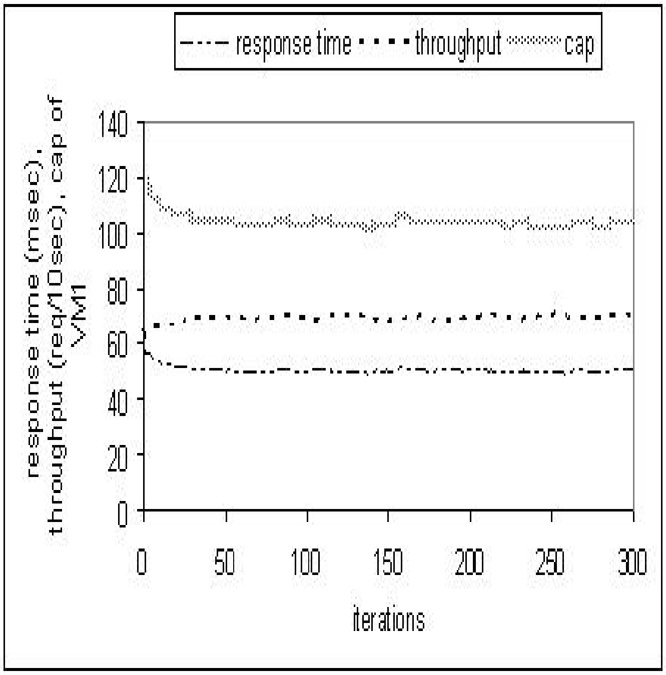


Figure 23 Performance evaluation of Optimal control in virtualized environment : setting-4

VII. RESEARCH HIGHLIGHTS

The main goals of our work are

- ➢ To monitor various performances issues in Server Virtualization.
- ➢ To Study the optimization process and various issues in analyzing the performance of Server Virtualization.
- ➢ To identify various parameters and issues for evaluating performance of virtualization in cloud computing environment in terms of CPU, memory performances.
- ➢ To study various issues to understand effectiveness of existing Quality of Service (QoS) controls on resource usage and thereby application performance.
- ➢ To design and implement a controller that optimizes the performance of applications running on guest domains.
- ➢ The goal is to dynamically compute the CPU shares for the virtual machine in such a way that the application through put is maximized, while keeping the response time as low as possible with minimum possible allocation of CPU share for the guest domain.
- ➢ To maintain the QoS of the applications running inside the VMs around some desired value. And also
- ➢ To minimize the resource usage by the application running inside the VMs while maximizing the application performance. This goal can also be called as optimal control.
- ➢ The goal is to consolidate the Data Center and increase its performance.

VIII. CONCLUSION

In this paper, we described the problem of delivering QoS to the applications running inside the virtualized environment. Our work focused on devising a mechanism for computing the share of the resources to be allocated to each virtual machine in such a way that desired QoS is delivered to the applications running inside virtual machines. We designed the feedback control system for virtualized environment. We designed and implemented controller, sensor, and capacity analyzer modules as a part of the control system. Sensors measure the QoS delivered by the applications. Controller uses these QoS values to decide new values of resource management parameters like cap of a virtual machine. Capacity analyzer verifies whether the resource demands of all applications can be fulfilled with the given physical server or not. We evaluated the performance of the proposed control system by deploying two tier applications in the virtualized environment test bed. We carried out the experiments with desired response time of the application as reference input and cap of the virtual machines in which application resides as the control input. We implemented the sensor for carrying out response time measurements at the servers. The results of the experiments shows that control system is able to set the values of cap accurately even in the presence of disturbance.

REFERENCES

[1] Anton Belglazov, Jemal Abawajy and Rajakumar Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing”, International Journal of Future Generation Computer Systems”, Elsevier publications, 755-768, 2012.

[2] Abirami S.P. and Shalini Ramanathan, ” *Linear Scheduling Strategy for Resource Allocation in Cloud Environment* “,International Journal on Cloud Computing: Services and Architecture(IJCCSA),Vol.2, No.1,February 2012.

[3] Abhinav Kamra, Vishal Misra, and Erich M. Nahum. Yaksha: “*A self-tuning controller for managing the performance of 3-tiered web sites”*. Proceedings of 12th International Workshop on Quality of Service(IWQoS), 2004.

[4] Andrew J. Younge, Robert Henschel, James T. Brown, Gregor von Laszewski, Judy Qiu, Geoffrey C. Fox, ”*Analysis of Virtualization Technologies for High Performance Computing Environments*”, Pervasive Technology Institute, Indiana University 2729 E 10th St., Bloomington, IN 47408,U.S.A.ajyounge,henschel,jatbrown,gvonlasz,xqiu,gcf }@indiana.edu. Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD. 9-16.

[5] Andrea Arcangeli, Izik Eidus, Chris Wright, ” *Increasing memory density by using KSM*”, Red Hat, Inc. aarcange@redhat.com, ieidus@redhat.com, chrisw@redhat.com.

[6] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya, ”*Energy–aware resource allocation heuristics for efficient management of Data Centers for Cloud Computing*”, Future Beneration Computer Systems(2012), Elsevier, also available at Science Direct, Sponsored by Cloud Computing and Distributed Systems(CLOUDS).

[7] Aravind Menon, Jose Renato, Yoshio Turner,G. (John) Janakiraman, Palo Alto john,Willy Zwaenepoel, ” *Diagnosing Performance Overheads in the Andrzej Kochut and Kirk Beaty. On strategies for dynamic resource management in virtualized server environments”*. MASCOTS 2007: IEEE / ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), October 2007.

[8] “*Xen Virtual Machine Environment*”, VEE’05, June 11-12, 2005, Chicago, Illinois, USA. Copyright2005ACM1-59593-047- 7/05/0006...$5.00.

[9] B.Thirumala Rao, N.V.Sridevi, V.Krishna Reddy, L.S.S.Reddy, ”*Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing* “, Global Journal of Computer Science and Technology Volume XI Issue VIII May 2011 .

[10] Bao Rong Chang, Hsiu-Fen Tsai, Chi-Ming Chen, ” *Evaluation of Virtual Machine Performance and Virtualized Consolidation Ratio in Cloud Computing System*”, Journal of Information Hiding and Multimedia Signal Processing c ◯2013 ISSN 2073-4212 Ubiquitous International Volume 4, Number 3, July 2013.

[11] Bhukya, D.P. ; Ramachandram, S. ; Reeta Sony, A.L ,“*IO Performance Prediction in Consolidated Virtualized Environments*”.

[12] Carl A. Waldspurger, ” *Memory Resource Management in VMware ESX Server*”, VMware, Inc. Palo Alto, CA 94304 USA carl@vmware.com, Proceedings of the 5th Symposium on Operating Systems Design and Implementation.

[13] Diego Ongaro Alan L. Cox Scott Rixner, “ *Scheduling I/O in Virtual Machine Monitors*”, VEE’08, March 5–7, 2008, Seattle, Washington, USA.

[14] Diogo M. F. Mattos, Lyno Henrique G. Ferraz, ” *Virtual Network Performance Evaluation for Future Internet Architectures*” JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 4, NO. 4, NOVEMBER 2012.”

[15] Diego Ongaro Alan L. Cox Scott Rixner, ” *Scheduling I/O in Virtual Machine Monitors*”, VEE’08, March 5–7, 2008, Seattle, Washington, USA. Copyright c 2008 ACM 978-1-59593-796-4/08/03...$5.00.

[16] Diwaker Gupta1, Ludmila Cherkasova, Rob Gardner, Amin Vahdat1,” *Enforcing Performance Isolation Across Virtual Machines in Xen*”, Enterprise Software and Systems Laboratory HP Laboratories Palo Alto HPL-2006-77 May 4, 2006*.

[17] Diwaker Gupta, Ludmila Cherkasova, Rob Gardner, and Amin Vahdat. “*Enforcing performance isolation across*

virtual machines in xen". Middleware 2006: Proceedings of ACM/IFIP/USENIX 7th International Middleware Conference, 2006.

[18] Himanshu Raj, Ripal Nathuji, " *Resource Management for Isolation Enhanced Cloud Services*", CCSW'09, November 13, 2009, Chicago, Illinois, USA. Copyright 2009 ACM 978-1-60558-784-4/09/11 ...$10.00.

[19] Horacio GonAlez Velez, Maryam Kontagora, "*Performance evaluation of mapreduce using full virtualization on a departmental cloud*", Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 2, 275–284 DOI: 10.2478/v10006-011-0020-3(AMCS).

[20] Indrani Paul, Sudhakar Yalamanchili, Lizy K. John, " *Performance Impact of Virtual Machine Placement in a Datacenter*".

[21] Leslie, I.M. McAuley, D. Black, R. Roscoe, T. Barham, P. Evers, D.Fairbairns, and R. Hyden."*Design and implementation of os to support distributed multimedia applications (nemesis)"*. IEEE Journal of Selected Areas in Communications, 1996.

[22] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. "*When virtual is harder than real: Resource allocation challenges in virtual machine based it environments. Technical report*", HP Laboratories Palo Alto., February 2007.

[23] nikolaus huber, marcel von quast, Michael Hauck, Samuel Kounev, "*Evaluating and modeling virtualization performance overhead for cloud environments*", CLOSER 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science, Noordwijkerhout, Netherlands, 7-9 May, 2011. SciTePress 2011 ISBN 978-989-8425-52-2.

[24] Nikolaus Huber, Marcel von Quast, Michael Hauck, Samuel Kounev, " *Evaluating and modeling virtualization Performance overhead for cloud environments*", *Journal of Information Hiding and Multimedia Signal Processing*ⓒ 2013 ISSN 2073-4212, **Ubiquitous International** Volume **4**, Number **3**, July **2013.**

[25] Paul Barham∗, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer†, Ian Pratt, Andrew Warfield, "Xen and the Art of Virtualization", SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA. Copyright 2003 ACM 1-58113-757-5/03/0010 ...$5.00.

[26] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauery, Ian Pratt, and AndrewWareld. "*Xen and the art of virtualization. nineteenth ACM symposium on Operating systems principles*", 2003.

[27] Pradeep Padala, Xiaoyun Zhu, ZhikuiWang, Sharad Singhal, and Kang G. Shin. "*Performance evaluation of virtualization technologies for server consolidation"*. Technical report, HP Laboratories Palo Alto., April 2007.

[28] Qingling Wang, Carlos A. Varela , "Impact of Cloud Computing Virtualization Strategies on Workloads 'Performance", Department of Computer Science Rensselaer Polytechnic Institute, Troy, NY, USA http://wcl.cs.rpi.edu/ {wangq9, cvarela}@cs.rpi.edu, *UCC, page 130-137. IEEE Computer Society,* (*2011*).

[29] Rahul Gundecha. "*Measurement-based evaluation of virtualization platforms"*. Technical report, Indian Institute of Technology, Bombay, april 2007.

[30] Ratul K. Majumdar, Krithi Ramamritham, Ravi N. Banavar, and Kannan M. Moudgalya. "*Disseminating dynamic data with qos guarantee in a wide area network: A practical control theoretic approach*". 10th IEEE Real-Time and Embedded Technology and Applications Symposium, 2004.

[31] Sajib Kundu, Raju Rangaswami, Kaushik Dutta, Ming Zhao, " *Application Performance Modeling in a Virtualized Environment*", School of Computing & Information Sciences, College of Business Administration Florida International University {skund001, raju}@cs.fiu.edu kaushik.dutta@business.fiu.edu,zhaom@cs.fiu.edu,

[32] Shicong Meng, Ling Liu, "*Monitoring-as-a-Service in The Cloud*", ICPE'13, April 21–24, 2013, Prague, Czech Republic. ACM 978-1-4503-1636-1/13/04.

[33] S. Keshav. "*A control-theoretic approach to flow control*". Proceedings of the ACM SigComm, 1991.

[34] Sriram Govindan, Jeonghwan Choi, Arjun R. Nath, Amitayu Das, " *Xen and Co.: Communication-Aware CPU Management in Consolidated".*

[35] "*Xen-Based Hosting Platforms*", 0018-9340/09/$25.00 2009 IEEE Published by the IEEE Computer Society.

[36] Sujay Parekh, Dawn M. Tilbury, Joseph L. Hellerstein, and Yixin Diao. "*Feedback control of computing systems*". John Wiley and Sons, Inc, 2004.

[37] Tarek Abdelzaher, Kang G. Shin, and Nina Bhatti. "*User-level qos-adaptive resource management in server end-systems*", IEEE Transactions on Computers, 52.

[38] Xiao Zhang Eric Tune Robert Hagmann Rohit Jnagal Vrigo Gokhale John Wilkes," *CPI^2: CPU performance isolation for shared compute clusters*", Google, Inc, 2013 ACM 978-1-4503-1994-2/13/04. $15.00.

[39] .Xue Liu, Xiaoyun Zhu, Pradeep Padala, ZhikuiWang, and Sharad Singhal. "*Optimal multivariate control for differentiated services on a shared hosting platform*". Proceedings of the 46th IEEE Conference on Decision and Control (CDC'07), December 2007.

[40] Ying Lu, Avneesh Saxena, and Tarek F. Abdelzaher. "*Differentiated caching services; a control-theoretical approach*". International Conference on Distributed Computing Systems, 2001.

[41] Zongjian He, Guanqing Liang, " *Research and Evaluation of Network Virtualization in Cloud Computing environment*", IEEE Third International Conference on Networking and Distributed Computing (ICNDC), 2012 at Hangzhou,40-45, ISSN :2165-5006,Print ISBN:978-1-4673-2858-6,INSPEC Accession Number:13263829,Digital Object Identifier :10.1109/ICNDC.2012.18.

[42] ZhikuiWang, Xiaoyun Zhu, Pradeep Padala, and Sharad Singhal. "*Capacity and performance overhead in dynamic*

resource allocation to virtual containers". Technical report, HP Laboratories Palo Alto., April 2007.

[43] "*Application Performance Management in a Virtualized Environment Growing* ", WHITE PAPER: APPLICATION PERFORMANCE MANAGEMENT.

[44] " *Better virtualization of XenApp and XenDesktop with XenServer*", XenApp and XenDesktop with XenServer White Paper.

[45] "*Experimental Evaluation of the Performance-Influencing Factors of Virtualized Storage Systems*", Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on Digital Object Identifier: 10.1109/ICCIC.2010.5705753, Publication Year: 2010 , Page(s): 1 - 4

[46] *VMware site. http://www.vmware.com/*.

[47] *Official Xen project site. http:/ /www.cl.cam.ac.uk/research/srg/netos/xen/.*

[48] *httperf. http://www.hpl.hp.com/research/linux/httperf/.*

[49] Website of Muffin proxy server. http://muffin.doit.org.

Authors Profile

Vedula Venkateswara Rao is a Ph.D Candidate in the Department of Computer Science Engineering at Gitam Institute of Technology, Gitam University, Visakhapatnam, and Andhra Pradesh, India. He received Masters Degree in Computer Science Engineering from Jawaharlal Nehru Technological University Kakinada, Masters Degree In Information Technology from Punjabi University, Patiayala, India. His research interests include Cloud Computing and Distributed Systems, Data Mining, Big Data and Image Processing. He published several papers in International conferences and journals.

Dr. Mandapati Venkateswara Rao is Professor in Department of Information Technology at Gitam Institute of Technology, Gitam University, and Visakhapatnam, India. He Has received M.Tech in CST and PhD in Robotics from Andhra University. His Research Interests includes Robotics, Cloud Computing and Image processing. He published several papers in International conferences and journals.

.