MONITORING, INTROSPECTING AND PERFORMANCE EVALUATION OF SERVER VIRTUALIZATION IN CLOUD ENVIRONMENT USING FEED BACK CONTROL SYSTEM DESIGN

[1]VEDULA VENKATESWARA RAO, [2]Dr. MANDAPATI VENKATESWARA RAO

[1]Research Scholar, Department of Computer Science Engineering, Institute of Technology, Gitam University, Visakhapatnam, India

Associate Professor, Department of Computer Science Engineering, Sri Vasavi Engineering College, Tadepalligudem, India

[2]Professor, Department of Computer Science Engineering, Gitam Institute of Technology, Gitam University, Visakhapatnam, India

E-mail: [1]venkatvedula2012@gmail.com, [2]mandapti_venkat@yahoo.co.in

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. 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. 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.

Keywords: *Cloud Computing, Data Center, Virtualization, hypervisor, Xen, virtual machine, Green IT, scheduler, performance, response time, throughput, feedback control theory.*

1. 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 [5] [36] 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]. 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[25][45] 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[6] [9] [12]. 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 [25] [45] 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[15] [16] 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 [17] [26] [27] [30]. 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 modeling the performance behavior of the applications residing in virtual machines. Client SLAs 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.

2. LITERATURE REVIEW

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 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 AMD 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 utilization and so on. Also during the past few years due to the popularity of virtualization and its utilization 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.

2.1 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, datacenter and so on [17].
- ➢ 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) [27].

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 [31] [44].

2.2 Virtualization of 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[7] 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[33] became the first company to release a virtualization product for x86 based machines which was named "VMware Virtual Platform". At present, VMware server [33], VMware ESX [30], XenServer [26], Microsoft Virtual server [32] 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 OSes 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.

2.3 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 datacenters. 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. 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.

2.4 Application Performance and Virtual Machines

The performance of an application should not get adversely affected by the other applications running on the same hardware. Gupta et al [11] 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.

2.5 Feed Back 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 [34]. The very famous invention of James Watt, the steam engine [34] 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 [34]. Some of the widely used applications of feedback control theoretic approach involves automobile cruise control, aircraft cruise control, temperature maintenance using thermostat[28][16][34]. 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.

3. 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.

3.1 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. Figure1 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 [45], Xen [46]. In our study, we have used open source VMM solution Xen [24] [25] [46].

• 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.

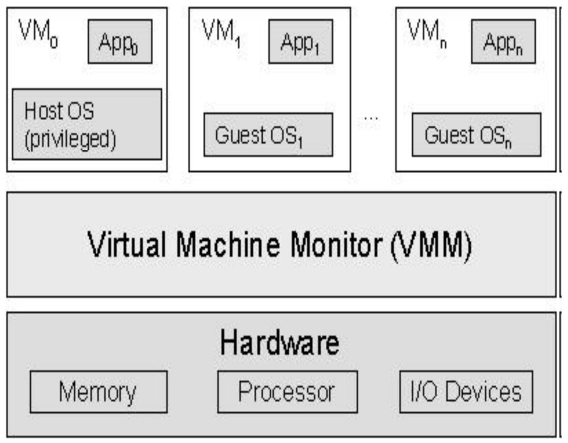


Figure 1: Virtualized Environment

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 OSes are like the normal processes running on it. The VMM interface is available in host OS through which guest OSes are managed. The term domain is alternatively used in place of virtual machine. Host OS is often called as Domain-0 where as guest OS are called DomUs.

3.2 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.

3.3 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. Figure2 pictorially depicts the scenario of virtual machine environment.

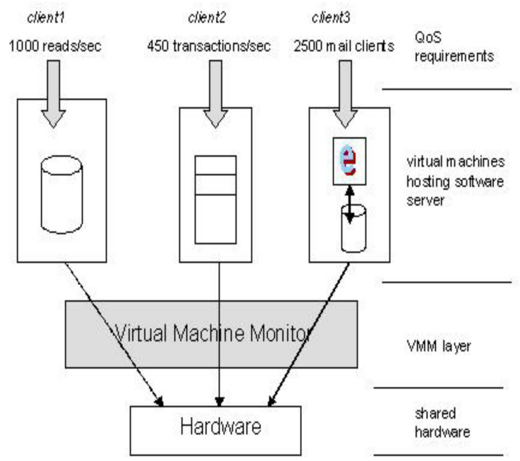


Figure 2: Applications Running Inside Virtualized Environment

Performance Isolation as described by [15] [16] 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.

Table 1: Effect On Mixed Load On The Performance Of Applications In Virtualized Environment

Statistics of web server running in virtualized environment						
	Weight	Cap	Load	CPU usage	Requests per sec	Transfer rate (Kbytes per sec)
Experiment1: With Web Server running						
Domain0	256	400	-	-	NA	
VM2	256	400	-	-	NA	
VM3	256	400	Web Server	180	797.61	1035.17
Experiment1: Mixed Load 1 VM CPU Load,1 With Web Server running						
Domain0	256	400	-	-	NA	
VM2	256	400	CPU	100	NA	
VM3	256	400	Web Server	180		

In above table 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. The experiment 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 [15] [16] [21] [26] [41] revealed that there is compelling need of having better performance isolation mechanism in Xen. This is also evident from the fact that three schedulers [46] 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.

3.4 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.

4. PROPOSED METHOD

In this section we present our mechanism i.e. “Scheduler based on feedback control system” 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 it more robust to changes in the operating environment.

4.1 Feed Back 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.

4.1.1 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.

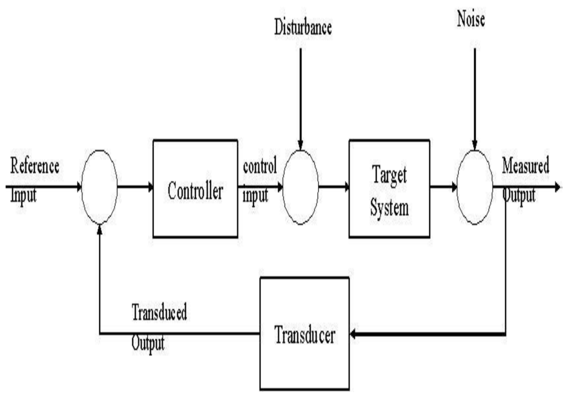


Figure 3: Typical Feedback control system

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.

4.2 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 shows the architecture of QoS (quality of service) aware virtualized environment. Datacenters host number of physical servers which are shared among multiple client applications.

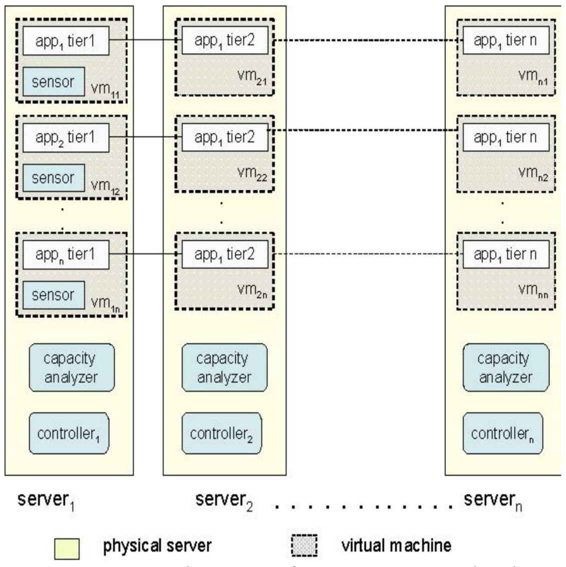


Figure 4: Architecture of QoS aware virtualized environment

As shown in above architecture 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.

4.3 Feed Back Control System Design

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 modeling 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.

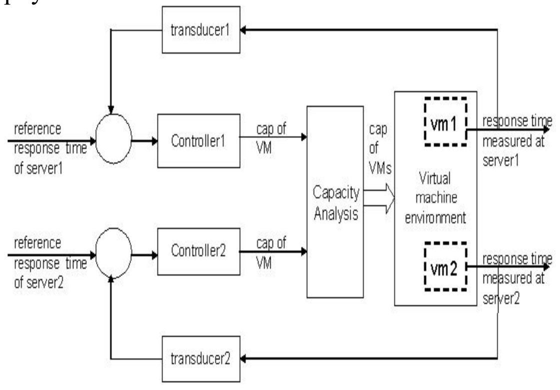

transducer1
reference response time of server1
Controller1
cap of VM
cap of VMs
Capacity Analysis
vm1
Virtual machine environment
response time measured at server1
reference response time of server2
Controller2
cap of VM
vm2
response time measured at server2
transducer2

Figure 5: Feedback control system for virtualized environment

As shown in the figure5, 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.

5. 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.

5.1 Components of Test Bed

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 6 explains the Testbed for QoS aware virtualized environment.

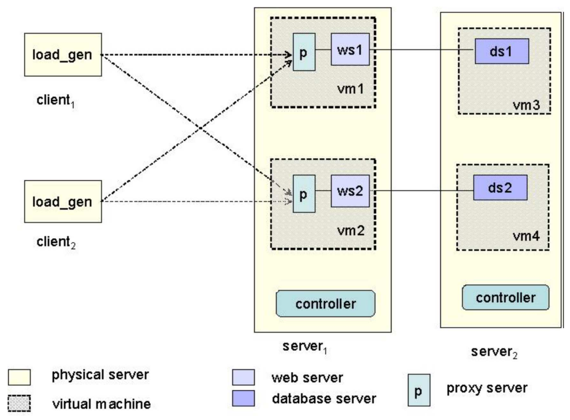


Figure 6: Testbed for QoS aware virtualized environment

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.

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 datacenter 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.

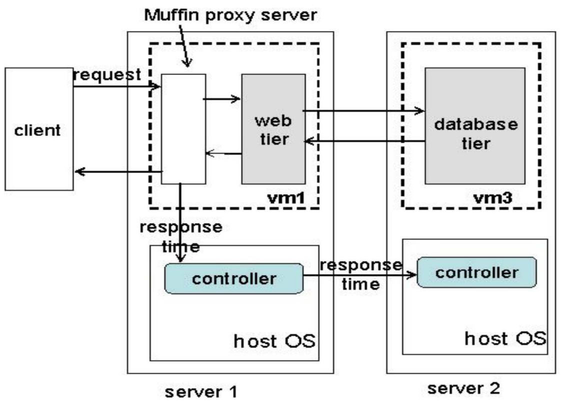


Figure 7: Response time measurement and flow of a request through the Testbed

The above 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.

5.2 Work Load 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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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.

The following diagram explains Optimal Control System architecture.

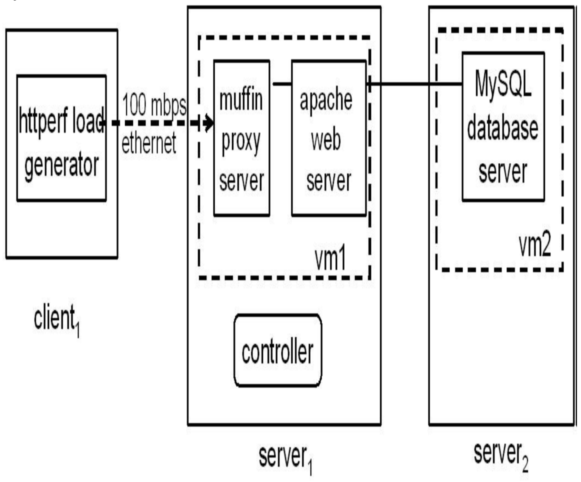


Fig 8 Optimal Control System design Architecture

6. PERFORMANCE EVALUATION

This section present the results of our experiments carried out to evaluate the feedback control system. For comparison purpose, we carried out a experiment without controller running. The cap of the virtual machines were 47 and 40 respectively which are the average values of cap set in the second experiment carried out with the controller running. The desired response time values are 180msec and 220msec for the two web servers respectively which are same as the second experiment with the controller which is described below. Graph in figure 9 shows the values of response time delivered by the web servers running in virtual machines. The table2 gives more clear picture of the results we got in this experiment. We classified the % error values in certain ranges. Generally any error over magnitude of 10% might not be tolerable by the clients. As shown in the table, 35% and 25% time's error values are having magnitude of more than 10%.

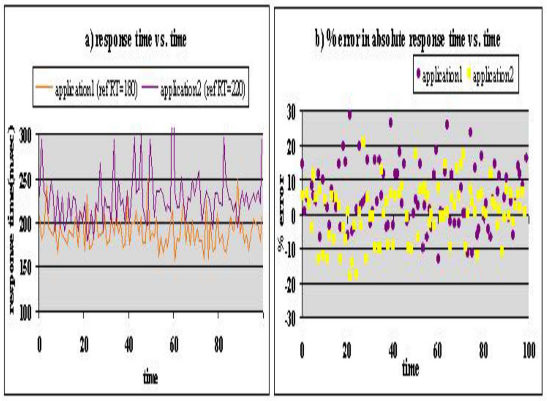


Figure 9: Evaluation of virtual machines without controller

Table 2 Summarized result of virtual machines without controller

Summarized Result							
% Errors	*< -10 %*	*-10 % to -5%*	*-5% to 0%*	*0 %*	*0% to 5%*	*5% to 10 %*	*> 10 %*
Response Time Values							
Application1	*3.1*	*11.4*	*12.3*	*4.5*	*20.2*	*16.4*	*32.1*
Application2	*14.8*	*11.3*	*15.9*	*4.6*	*25*	*18*	*10.4*

Figure 10 shows plots of the values of the response time delivered by the both web servers running in the virtualized environment. This experiment was carried out in the same environment as of first experiment without control. The reference inputs given for this experiment were 180msec and 220msec for the two web servers respectively.

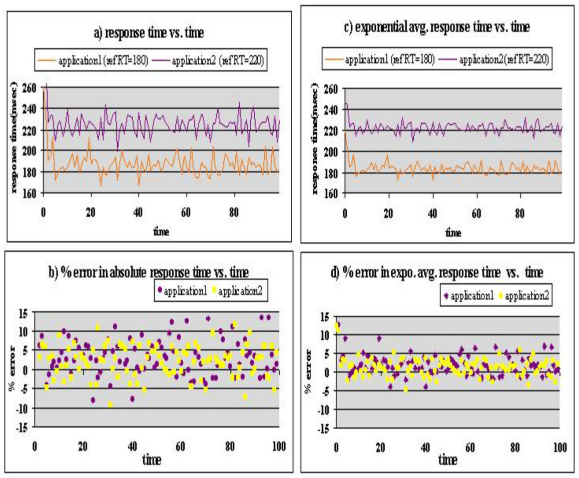


Fig10 Evaluation of the feedback controller

The graph a) shows the graph of response time values delivered by the web servers against the time. Graph shows that response time values are generally very close to the desired values of response time. Graph b) of % error against the time gives the values of errors between actual and desired values of the response times. The graph c) of exponential average response time against time gives the plot of the response times which are being produced by transducer which are further used by the controller in the computation of new cap value. This experiment was done with exponential factor of 0.2
As shown in the table 3, application1 delivers response time with error of magnitude less than 10% for 86% of times whereas application2 delivers response time with error of magnitude less than 10% for 89% times. In the first experiment these values were 65% and 75% respectively. This shows that the controller is able to deliver the desired response time. Both the applications deliver the response time with error magnitude of less than 5% for around 60% of time each. Table also lists out the error values when exponentially averaged response time is compared with the reference response time. In this comparison, we got only 4% and 2% error from application1 and application2 respectively.

Table 3 Summarized Result Of Evaluation Of Feedback Controller

Summarized Result							
%Errors	<-10%	-10% to -5%	-5% to 0%	0%	0% to 5%	5% to 10%	> 10%
Response Time Values							
Application1	0	2	12	10	38	24	14
Application2	0	0	18	8	48	16	11
Expontial Average Response Time Values							
Application1	0	0	10	14	62	10	4
Application2	0	2	16	16	56	8	2

To illustrate the robustness of the feedback controller, we carried out the experiments in presence of disturbance. We start executing a thread periodically on the virtual machine where application1 is running i.e. on vm1. This thread is CPU hogging loop which alternately sleeps and executes some computation. This is explained in Fig 11.

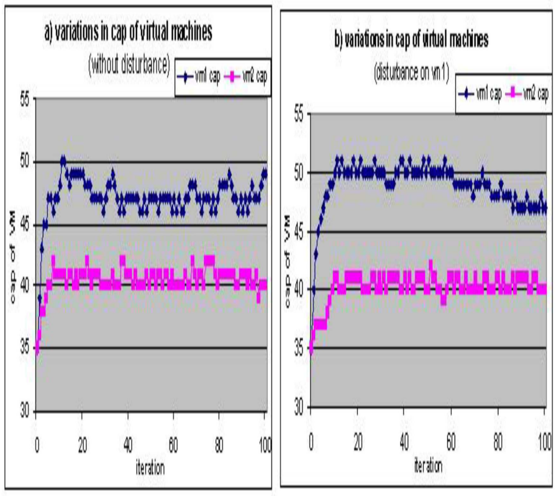


Figure 11: Evaluation of the feedback controller in presence of disturbance

As shown in the table 4, application1 delivers response time with error of magnitude less than 10% for 85.75% of the times, whereas application2 delivers response time with error of magnitude less than 10% for 86.5% times. Both the applications deliver the response time with error magnitude of less than 5% for around 50% of time each. The error values when exponentially averaged response time is compared with the reference response time are 0.73% and 0% for application1 and application2 respectively. Hence these results show that our controller is robust enough in presence of the disturbances.

Table 4 Summarized Result Of Evaluation Of Feedback Controller In Presence Of Disturbance

Summarized Result							
%Errors	<-10%	-10% to -5%	-5% to 0%	0%	0% to 5%	5% to 10%	> 10%
Response Time Values							
Application1							
Application2	0.46	2.0	11.6	5.38	29.6	34.5	16.2
Exponential Avg, RT Values	0.03	3.4	20.0	6.27	29.1	27.5	13.5
Application1							
Application2	0	0.2	8.87	18.0	68.8	3.45	0.73

The graphs in figure 12 shows the values of cap set by the controller. Graph a) shows the values of cap in first experiment which was carried out without any disturbance whereas graph b) shows the values of cap from second experiment. From these graphs, we can infer that curve of cap of virtual machine vm1 in second experiment is relatively shifted upwards than the corresponding curve in first experiment. This happens because the virtual machine vm1 has extra load in terms of a thread running periodically which represent disturbance in the system. Average values of cap in the first experiment are 46.98 and 40.48 for vm1 and vm2 respectively, whereas average values of cap in the second experiment are 48.85 and 40.18 for vm1 and vm2 respectively. This shows that in second experiment there is 3% increase in CPU demand by virtual machine vm1 due to disturbance.

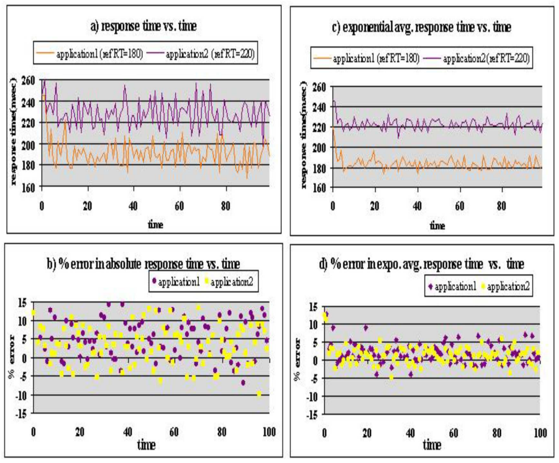
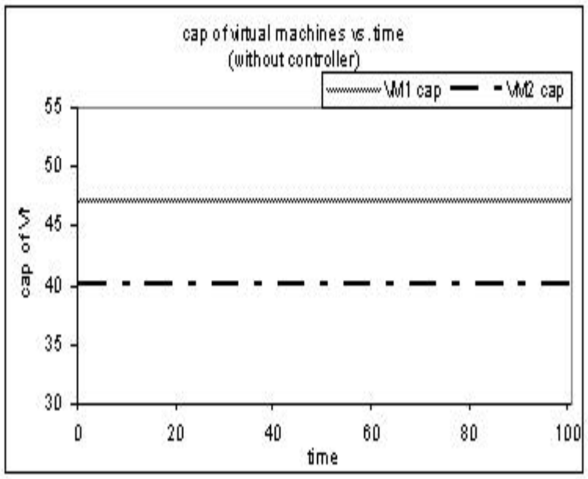


Figure 12: Variation of CAP Values of Virtual Machine.

The graph in the Figure 13 shows the values of cap in the Experiment, which had no controller running in the system. Graph in the Figure 6.9 shows the values of cap in Experiment which was carried out without any disturbance whereas graph in the Figure 6.10 shows the values of cap from Experiment in presence of disturbance. From these graphs, we can infer that curve of cap of virtual machine VM1 in Experiment with controller in presence of disturbance is relatively shifted upwards than the corresponding curve in Experiment with controller. This happens because the virtual machine VM1 has extra load in terms of the thread running periodically which represent disturbance in the system. Average values of cap in the Experiment with controller are 46.98 and 40.48 for VM1 and VM2 respectively, whereas average values of cap in the Experiment 3 are 48.85 and 40.18 for VM1 and VM2 respectively. This shows that in experiment with controller there is 3% increase in CPU demand by virtual machine VM1 due to disturbance.

Figure 13: CAP of virtual machines running without controller

The Graphs in Figure 14 & Figure 15 explains the



variation in CAP of Virtual Machines without noise and in presence of noise.

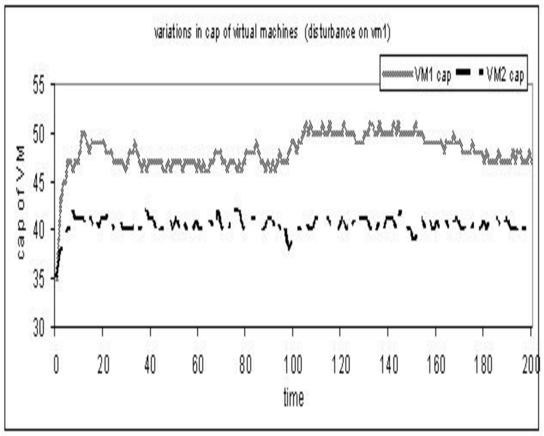


Figure 14: Variation in CAP values of Virtual Machine

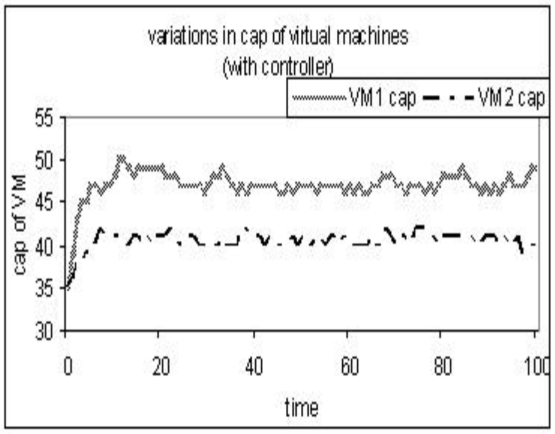


Figure 15: Variation in CAP values of Virtual Machine in presence of disturbance

7. RESEARCH HIGHLIGHTS

The main goals of our work are

1. To monitor various performances issues in Server Virtualization.

2. To Study the optimization process and various issues in analyzing the performance of Server Virtulization.

3. To identify various parameters and issues for evaluating performance of virtualization in cloud computing environment in terms of CPU, memory performances.

4. To study various issues to understand effectiveness of existing Quality of Service (QoS) controls on resource usage and thereby application performance.

5. To design and implement a controller that optimizes the performance of applications running on guest domains.

6. 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.

7. To maintain the QoS of the applications running inside the VMs around some desired value. This goal can be called as Reference tracking.

8. 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.

9. The goal is to consolidate the Data Center and increase its performance.

Feedback control system acts as scheduler who monitors the performance of processes running on different VM's and according to feedback it received it optimizes the allocation of CPU to different processes so that QoS is served for all the processes. The Results from figures 9, 10, 11 shows the role of feedback controller in allocating CPU to different virtual machines in optimizing manner. The Figures 13, 14, 15 shows how Scheduler based on Feedback control system changes cap values for keeping optimization of CPU utilization.

8. 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.

REFRENCES:

[1] 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.

[2] 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.

[3] 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.

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

[5] 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).

[6] 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.

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

[8] 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 .

[9] 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.

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

[11] 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.

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

[13] 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."

[14] 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.

[15] 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*.

[16] 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.

[17] 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.

[18] 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).

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.**

[24] 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.

[25] 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.

[26] 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.

[27] 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*).

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

[29] 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.

[30] 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,

[31] 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.

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

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

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

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

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

[37] 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.

[38] .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.

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

[40] 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.

[41] 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.

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

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

[44] "*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
[45] *VMware site. http://www.vmware.com/*.
[46] *Official Xen project site. http://www.cl.cam.ac.uk/research/srg/netos/xen/.*
[47] *httperf. http://www.hpl.hp.com/research/linux/httperf/.*
[48] *Website of Muffin proxy server. http://muffin.doit.org/.*