UiO : Department of Informatics
University of Oslo

Control System Development using a PID Controller Mechanism for Coordinating Vertical and Horizontal Elasticity

Shaikh Mohammad Farabi

Master's Thesis Spring 2017

Control System Development using a PID Controller Mechanism for Coordinating Vertical and Horizontal Elasticity

Shaikh Mohammad Farabi

22nd May 2017

Acknowledgements

First thanks to God himself that I have finished the research in time. I am so much grateful to University of Oslo(UiO) and Oslo and Akershus University College to give me an opportunity to provide me such a nice environment . Also thanks to my lovely supervisors Desta Haileselassie Hagos and Ashish Rauniyar for all sort of assistance during the thesis period. Many thanks to the Professor Anis Yazidi and Harek Haugerud.Many many thanks to all the faculty members of UiO and HIOA. At last thanks to my beloved family and friends.

Abstract

The popularity of the cloud computing has been increased exponentially in recent times. The rapid development of cloud computing technologies has lead to a paradigm shift in the way computing resources are provisioned. Huge number of servers are deployed to serve emerging number of end users everyday. Therefore, the complexity arises to maintain the huge workload as well as to provide expected level of services. Web based applications should have to ensure desired level of quality of services despite dynamic and continuous changes of workload.

The main focus of this thesis work is to propose a control system which is implemented with a feedback controlling mechanism known as Proportional Integral Derivative (PID) controller with an existing hybrid controller to dynamically allocate the resources for interactive and non-interactive applications. Traditional approaches are used to take elasticity based decisions such as either monitoring the resource usage or just merely based on Quality of Services (QoS) of solely latency based critical applications specially the webservers. Furthermore, our focus also lies on the batch processing software which is different than the traditional latency critical applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud computing opens up a new era in the field of computing. The popularity and usability of cloud computing is increasing day by day because it provides numerous facilities to the users like flexibility, reliability, elasticity and infinite computing resources and many more[73]. Therefore, many companies and organizations, either big or small are getting benefit from this technology. They are moving their business to cloud infrastructure. A survey conducted by Rightscale in the beginning of 2016 shows that more enterprise shift their workloads to cloud, especially on private cloud. Moreover, within a year almost 17 percent of enterprises now have more than 1000 VM's in public cloud, up from 13 percent while 31 percent of enterprises running more than 1000 VM's in private cloud as compared to 22 percent in 2015[78]. Similarly, another article published on Forbes around 2016 revealed that spending on public cloud infrastructure- hardware and software is forecast to reach 38 billion USD, growing to 173 billion USD in 2026[17]. The entire statistics clearly reveals that how fast cloud computing infrastructure is growing bigger and getting popular among the organizations and end users these days.

Cloud computing also create a new dimension and rises several concepts and technologies like data center and hardware virtualizations[22]. Data centers contributes a major role on cloud computing and it has received significant attention as a cost-effective infrastructure for storing large volumes of data and moreover, hosting large-scale service applications.[9]. The main benefit of data centers is that it allows dynamic resource allocation across large server of pools and it is very much agile and cost effective[32]. Almost all of the large multinational companies for example Amazon, Youtube, Google, Facebook, Ebay and Yahoo use data centers for multipurpose functions like storage, web search, and large scale computations[80][31]. The rising demand and popularity of cloud service and applications hosting in data centers a multi-billion dollar business that plays a crucial role in the future Information Technology (IT) industry[9]. Data centers consists of almost tens of thousands of computers with significant aggregate bandwidth requirements[23]. Therefore, those large amount of computer within a data centers consume a large amount of energy. In a report published in US shows that only in US, the data

centers energy consumption rate in 2011 was approximately 100 KWh which is equivalent to 7.5 billion US dollar.[46]. Another statistics revealed that energy consumptions of data centers almost tripled within a decade, an average growth rate increased by 12 percent from 2000 to 2010[76]. Servers that are used in data centers to run applications resulting poor server utilization and high operational cost[9]. Data centers needs to be more energy efficient and environment friendly as well as they should be capable of handling dynamic hasty workloads. Therefore, dynamic resource provisioning is necessary not only to avoid the application performance degradation, but also to effectively and efficiently utilize resources. Furthermore, the infrastructures should have the capability to allocate resources according to the applications demands, which will be able to create better impression to the user and attract the users towards the services. Moreover, data centers at present use static resources which leads to a huge operational cost for organizations also provide poor service performances which kills userâs satisfaction. Most importantly in recent days the customer satisfactions is the most important scenario, and it is directly proportional to the business revenue which indicates lower customer satisfaction decrease business revenue. Recent survey reveals that one of the biggest electronic commerce and cloud computing company 'Amazon' found a page load slowdown of just 1 sec could cost 1.6 billion dollar in sales every year[21]. Another research conducted by Natural Resources Defense Council(NRDC) reveals that the average server operates at not more than 12 to 18 percent of its capacity while still drawing 30 to 60 percent of maximum power. Moreover, virtually idle servers consume power 24/7, which adds up to a substantial amount of energy usage[55].

Resource elasticity is an autonomic scaling platform and one of the most selling point of cloud computing[77][41][24]. Resource elasticity is defined as an elastic cloud which dynamically adapts resource allocation according to the consumer's choice. Moreover, resource provision depends on how much money users has paid for the service[77]. Resource elasticity are of two types: vertical elasticity and horizontal elasticity. Horizontal elasticity is defined as adding or removing virtual machines(VM) to or from application depending on the number of end users[45]. Furthermore, horizontal elasticity needs support from the application, for instance, to close and synchronize states among VMs but it does not require any extra support from the hypervisor. Therefore, it has become more popular and has been widely adopted in public clouds[45][67]. However, vertical elasticity considered as adding or removing resources such as CPU and memory to or from an individual VMs in order to cope with runtime changes[45]. Vertical elasticity has been achieved by changing the size of VMs. Horizontal elasticity is coarse-grained, because of the VMs in horizontal elasticity has static and fixed size configuration, which can be used or run for longer period. However, vertical elasticity is known as fine-grained because in vertical elasticity individual elements allocated to VMs such as CPU and memory can be dynamically changed for as short as a few seconds[45][41][67].

1.1 Problem Statement

The main objective of service provider or infrastructure provider is to provide high quality of service (QoS) to their customer at any time and maintain stable infrastructure while customer, on the other hand, want their application running 24x7 hours without any disturbance which indicates that they want their applications keep running(up) in any kind of obstacle or situations i.e 100 percent QoS is always in demand for customers. Now a days, most of the big companies and organizations transfer their business on cloud running very important applications and doing big transactions through service providers like Amazon. Therefore, a high QoS is very important for them. Even a single second of latency or disturbance or down on service can cause a massive loss. Here is some interesting facts about net user. In America, one in four people abandons surfing to a website if its page takes longer than four seconds to load. Similarly, 4 in 10(almost 50 percent) Americans give up accessing a mobile shopping site that would not load in just 3 seconds [21]. However, infrastructure providers are unknown to the pattern of web request. They have no idea about what kind web applications running on their infrastructure and the variations of web spikes it may receive. In such situation, real-time applications for instance, latency-critical and even a small amount of interference can cause significant Service-Level Agreement (SLA) violations. Therefore, cloud service provider should have a dynamic infrastructure (resource allocation) to serve unpredictable spikes in user access. Otherwise, they may lost their clients. In this case, it is very essential to have deep investigation and research on how to provide better resource allocations on cloud platform specially on big data centers, to design and build a better controller for getting comparatively fair result. Furthermore, it is by far most important to have deep looks on which scaling is better, cost saving, easy to maintain from data center point of view.

The aim of this research is to propose a PID controller logic mechanism. The PID controller has to be designed in such a way which can integrate the existing autonomic controller and increase the performance of the controller.

The problem statement which is the basic foundation of this thesis has been divided into three parts.They are as follows:

1. *How can we determine a threshold value which is known as set value?*
2. *How can we build a dynamic PID controller considering the differences between the set point and measured value, and apply the correlation based on proportional, integral, and derivative terms?*
3. *How can we design and build a sophisticated Service Level Agreement (SLA) driven infrastructure to provide efficient QoS by applying better resource allocation?*

1.2 Expalnation of Terms:

1.2.1 Set Value:

Maximum desired response time will be set as an threshold value, which will be known as set value.The research work carried out by Ahmad et and al. [3] calculated the range of response time. In this paper, work the maximum desired response time is regarded as set value and it is described in the following section about it more precisely.

1.2.2 PID controller:

The goal of this paper work is to develop a proportional integral derivative controller or PID for an autonomic provisional controller . The autonomic provision controller has been made with the help of the control theory. The motivation to design the controller model and design feedback loops to make the cloud infrastructure dynamics(also known as self adaptive) to gain a proper and better balance between fast reaction and better stability. The main aim of the PID controller is to calculate the error value on the basis of set value and measured process value. Afterwards, the correction will take place with the basis of proportional, integral and derivative which also known as "Modes of control"[58][59] The controller provision of high Quality of Service(QoS) service by allocating the available resources based on heterogeneous requests that comes to the server.

1.2.3 SLA based Infrastructure:

The SLA based infrastructure indicates that if the measured value is more than the desired or it goes beyond the desired level then it is going to violate the overall goal of the research work. Therefore, we need to build a control system which can manage the code of SLA and perform better QoS by aloocating proper resources. In addition control systems must aware of redundant use of resources which can also vioates the QoS terminology.

1.3 Thesis Workflow

Chapter 2 we are discussing about the basic idea about Cloud computing models, cloud deployment models, hypervisor,type 1 and type 2 hypervisor, Xen hypervisor, KVM, Qemu , Libvirt Control Theory and PID Controller. Also some of the relevant papers and their work has been discussed at the end of the chapters.

Chapter 3 we are discussing about the methodology and how the idea will be developed to implement the control system based on our research questions.

Chapter 4 we are going to describe the controller characteristics controller models and how controller can perfrom the operations.

Chapter 5 we are going to highlight the most important part of the overall research.It includes the design and implementation part.

Chapter 6 we are going to focus the analysis part, the results will be discussed precisely and graphs has been made for complete analysis.

Chapter 7 we are going to reach a final decision of the research part which is consists of discussion, limitation and the future plan of this research.

Chapter 8 we are finally draw the end point of the entire research.

Chapter 9 consists of the Appendix portion where we put all the codes of the control systems.

Chapter 2

Background

In this chapter we are going to discuss about the basic concepts of cloud computing, virtualization , Type 1 and Type 2 hypervisor, KVM, Qemu. CPU schedulers.In addition vertical and horizontal scaling. All the topicsthat are essential to implement our research work. At the end of this chapter we represented some related works which are related to our research work also evaluate them in proper manner.

2.1 Concept of Cloud Computing:

Cloud computing is a combination of virtualization, storage, connectivity and processing power[48].Cloud computing can be defined as an on demand network access to a shared pool of network servers,storage,applications and services altogether[50].The term "Cloud" derived from the virtual private network which is known as VPN when telecommunication providers used VPN services for data communications [43]. Cloud can be structured and released with low management effort or service level of interaction[50].The demand of cloud computing rapidly increasing because it offers some true benefits e.g fast deployment, pay-foruse, lower costs, scalability, rapid provisioning, rapid elasticity,ubiquitous network access, greater resiliency, hypervisor protection against network attacks, low-cost disaster recovery and data storage solutions, on-demand security controls, real time detection of system tampering and rapid re-constitution of services [72].

2.2 Cloud Service Models:

Cloud computing has some user level agreements which describes how cloud services are made available to the clients[30].According to the usage of cloud services three basic service models are well known to the cloud service providers which includes Software as a Service(SaaS),Infrastructure as a Service(IaaS) and Platform as a Service(PaaS).The service models are briefly highlighted in the following sections.

2.2.1 Software as a Service(SaaS):

Software as a Service or SaaS known as "on demand Software" model in which applications are hosted by service provider or vendor and made available to users over a particular network through Internet.[1] In spite of the applications are running in a cloud Infrastructure but it is accessible from various client devices[50].One of the main advantage of Softaware as a Service that the Application Programming Interface or API allows the user to use the services without having no concern about the data storage and disk space.Disk space and data management will be done by the cloud service providers.[11].In spite of the main disadvantages of Software as a Service it is very strenuous to the user to ensure the proper security measures are in place as well as it is not guaranteed that the applications will available when it will be necessary.[72].The most notable key providers of SaaS software for instance Salesforce.com,Netsuite,Oracle,IBM, Facebook,Google Docs,Gliffy and Office 365.[40].

2.2.2 Infrastructure as a Service(IaaS):

Infrastructure as a Service or IaaS which basically provides generic function for hosting and provisioning of access to raw computing infrastructure and operating middleware software of its own[61].Infrastructure as a service(IaaS) has the capability to provide hardware,software,servers,storage and other infrastructure components to their own end user.[2].Moreover IaaS provision access,deploy and run arbitrary software to their own customer.[51].The main advantage of Infrastructure as a Service or Iaas has highly scalable resources which is adjusted on demand. In spite of the advantages,the main drawback of IaaS is that it so expensive and the service providers do not have any responsibility for backups, all backups has to be done by the clients' side.[3].The most notable IaaS providers are Amazon Web Services(AWS),Windows Azure,Google Compute Engine,Reckspace Open Cloud and IBM SmartCloud Enterprise.

2.2.3 Platform as a Service(PaaS):

Platform as a Service or PaaS is relatively new idea of cloud computing services.Platform as a Service provision platform which allows clientsto develop,run and manage the infrastructure without having any complexity to develop or relaunching the applications.[4] PaaS is basically create a bridge between hardware and application by abstract the infrastructure and support a set of application program interface(API) to cloud applications[29].The main advantage of Platform as a Service or PaaS is that it

[1]http://searchcloudprovider.techtarget.com/definition/cloud-services

[2]http://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS

[3]http://aiasecurity.com/2015/09/10/advantages-and-disadvantages-of-saaspaas-and-iaas/

[4]https://en.wikipedia.org/wiki/Platform$_{a}s_{as}ervice$

provides itś own infrastructure to the customers so that users do not have to install any in-house hardware and software to develop or run a new software.[5]

2.3 Cloud Deployment Models:

Besides cloud service models, cloud environment has another type of models which is known as cloud deployment models.Cloud deployment model is primarily differentiated by ownership,access and size[6].There are four cloud deployment models they are known as *Public Cloud, Hybrid Cloud, Community Cloud* and *Private Cloud*.

Public Cloud:
Public Cloud is the most well known cloud computing deployment model in which service providers make their services e.g. storage and applications publicly accessible over a network[7].In public cloud the cloud environment is owned by a third party vendor and they dynamically provision their resources to the customers'.[11].Amazon Web Services and Google Compute Engine are the most common example of public cloud.

Private Cloud:
Private Cloud has the same features like public cloud but it offers services through a proprietary architecture over a private network.Private cloud is owned and managed by a single organization or third party vendor.Private cloud has the potentiality to involve in virtualize business environments as well as it reevaluate decisions about existing resources.[8].Mirosoft,Eucalyptus,VMware is the common example of private cloud.

Community Cloud:
Community cloud is another cloud deployment model which shares infrastructure among several organizations.The organizations also share common concerns for example security,compliance and jurisdiction and make the access limited to the cloud consumers.The membership of community cloud is owned by the community members or third party vendor.Membership of the community does not ensure to gain control of the clouds entire IT resources[9].Outsiders does not get the access rights until or unless they will be approved by the community.

[5]http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS
[6]http://whatiscloud.com/cloud$_{d}eployment_{m}odels/index$
[7]http:/searchcloudcomputing.techtarget.com/definition/public-cloud
[8]https://en.wikipedia.org/wiki/Cloud$_{c}omputingPublic_{c}loud$
[9]http://whatiscloud.com/cloud$_{d}eployment_{m}odels/community_{c}louds$

Hybrid Cloud:
Hybrid cloud structure is different than other cloud computing deployment models because it consists of two or more different cloud deployment models.Basically hybrid cloud is a combination of other cloud computing models e.g. private cloud,public cloud and community cloud.Hybrid cloud is used by a large organization for their privacy for instance A is a large company has several confidential information they handle it with private cloud and for normal traffic and other situations they will manage public cloud.[11]

2.4 Concept of Virtualization:

Virtualization can be defined as the capability to run several virtual machines to single resources. The history of virtualization is not new at all.The idea of virtualization first came to 1960.IBM is the first company who came up with an idea of CP(Control Program) and CMS(Control Monitor System).CP acts as a host computer and it operates the VIrtual Machines which runs the guest computers known as CMS.[36]

2.5 Types of Virtualization:

Virtual machine has created over existing operating system and hardware known as hardware virtualization. In basis of hardware virtualization virtualization is divided in three parts.They are *Full Virtualization,Partial Virtualization* and *Para-Virtualization*.

2.5.1 Full Virtualization:

In full virtualization technique hypervisor allows to run multiple operating system to run on host operating[10].Basically in this technique the guest operating system completely unaware that it is running in a virtual environment.Therefore guest does not have the right to execute their own commands,they have to depend on the commands issued by the host operating systems. [11]

2.5.2 Para-Virtualization:

In para virtualization technique guest operating systems are modified so that it knows that it is running above a virtualized environment.[12] Also paravirtualization offers an interface that is similar to its underlying hardware.[13]. Furthermore it initiates hypercalls which makes explicit calls

[10]https://www.conres.com/it-products-solutions/news-events/three-types-of-server-virtualization-and-whats-best-for-you/

[11]https://www.quora.com/What-is-full-virtualization-partial-virtualization-and-paravirtualization

[12]www.udacity.com

[13]www.techopedia.com

to the hypervisor.The hypercall acts like system call in the operating system contains information of packages as well as it issues the desired state of hypercalls so that it can trap the VMM.[14]

2.5.3 Partial Virtualization:

In partial virtualization multiple instances of hardware are virtualized.Alike full virtualization the entire operating cannot be virtualized in partial virtualization.[15] However the most important thing is address space virtualization,which means each vm can be assigned with an individual address spaces. But the most notable drawbacks of partial virtualization is in some situations it needs backward compatibility.

2.6 Hypervisor

Basically hypervisor is a software layer which implements Virtual Machine(VM) and has the same architecture set like hardware[10]. Virtualization has been done by hypervisor a low level program that can run multiple operating system simultaneously.On the basis of hypervisor the system runs into virtual machines[26].Basically hypervisors are two kinds.[26]

Type 1 hypervisor known as bare metal hypervisor directly connected with system hardware and VMâs run onto it.All operating systems are running inside the virtual machines.Type 1 hypervisor provides hardware virtualization.

Type 2 hypervisor known as hosted hypervisor, connected with the host's operating system,and it supports other guest operating systems running above it. In this case the system is completely dependent on base operating system.

Type 1 hypervisor gives the better performance and flexibility than the type 2 hypervisor.Because it runs directly on the underlying hardware therefore it can operates a thin layer which provides resources to the VMâs.[16] On the other hand type 1 hypervisor is smaller than type 2 hypervisor.[17]

2.7 Xen Hypervisor

Xen is a well known and open source hypervisor which made up with a small software layer.In virtualization Xen hypervisor is very important because it executes several virtual resources for example

[14] www.udacity.com

[15] www.wikipedia.com

[16] http://searchservervirtualization.techtarget.com/tip/Virtualization-hypervisor-comparison-Type-1-vs-Type-2-hypervisors

[17] http://www.golinuxhub.com/2014/07/comparison-type-1-vs-type-2-hypervisor.html

vCPU,vMemory,event channels and shared memory of the hardware systems.Moreover it controls the I/O along with the memory access of the particular devices.[64].Xen has the capability to host the operating systems for instnace Windows and Linux also hosted simultaneously with a very less overhead performance that makes Xen a strong and effective hypervisor[8].Xen was a research project of University of Cambridge which was conducted by Ian Pratt and Simon Crosby[18].Later, Xen project development conducted by Citrix in the late 2007.Xen supports ARM various functionality such as ARM instruction sets,x86-64 and IA-32[38].

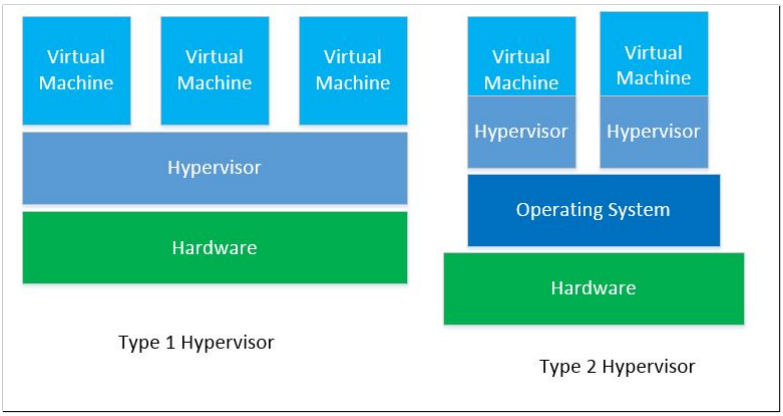

Figure 2.1: Type 1 and Type 2 Hypervisor

Xen-Architecture Xen is a bare metal hypervisor and it has some unique features such as small footprints and interface,paravirtualization,operating system agnostic and paravirtualization[19].

- **Small footprints and interface:** The total size of footprints around 1 MB.Footprints are smaller because Xen project uses small micro kernel design.Therefore interfaces are also limited to the guest make it more robust and secured.
- **Operating System agnostic:** The virtual machines of Xen are known as domains and they are referred as Doms[42].One special type of domain which known as domain0 or *dom0* [64]. Domain 0 or *dom0* is the main control stack which can run any paravirtualized operating systems like Linux.The guest operating systems which are running are optimized by changing their source code and replace instructions which mentioned as hypercalls [38].
- **Driver Isolation:** One of the most important part of Xen hypervisor is to allow the device driver to run inside a virtual Machine.If driver of a particular VM has crashed then it reboots or restart independently without any impact of the system.

[18]https://en.wikipedia.org/wiki/Xen
[19]https://wiki.xen.org/wiki/Xen$_{P}roject_{S}oftware_{O}verview$

- **Paravirtualization:** The main theme of paravirtualization is modifying the guest Operating Systems(OS).The main key factor of Xen hypervisor is that it implements the paravirtualization to modify the operating systems and make them faster and effective.It often faster than Hardware assistant Virtualization(HVM).

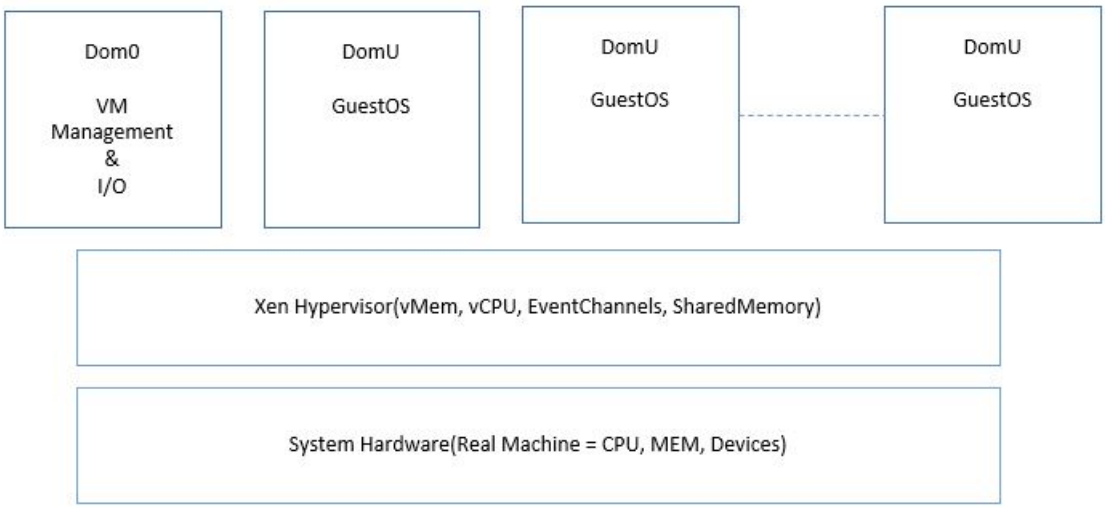

Figure 2.2: The Architecture of XEN

Whereas Xen is a bare metal hypervisor it directly runs on the hardware layer and directly handles CPU,Memory and interrupts.The Xen hypervisor architecture has also some features e.g. Guest Types, Domain 0 and Toolstacks.In Guests Types Hardware assisted Virtual Machine(HVM) guests and paravirtualized guests are allowed to run in Xen hypervisor.Domains are used to get limited hardware I/O access except one single domain which is known as *domain 0*[27][42]. Dom0 is the most significant and only privileged domain that has the capability to handle all inputs/outputs(I/O) [42].The user domains(dom0,dom1,dom2....domU) have played a key role to illustrate network drivers,data flows and control flows [42].If user domains need to access to the CPU,Memory or any I/O devices then it will forward request to the *dom0* [27]. Xen project stack and implications uses various frontend toolstacks which also play a great role in Xen project architecture.

2.8 KVM

Kernel based Virtual Machine which known as KVM is a virtualization infrastructure which is developed in Linux by Qumranet and later it turned into a type-II hypervisor[26, 85].QEMU emulator has been used to create KVM and it uses all operation and management tools of QEMU[26].The main speciality of KVM is unlike XEN it can run unmodified operating

systems like Linux and Windows Image.Also KVM has the capability to run multiple virtual machines on itś layer [20].Moreover the main advantage of Kernel Based Virtual(KVM) machine is that it can add virtualization capabilities to a standard Linux Kernel and turn into a hypervisor[84, 85].KVM is simple implementation rules which makes it stronger in Performance, Scalability, Security,Memory Management and Live Migration[84].KVM has simply two components one is **Kernel Space Device Driver** and another one is **User Space Emulator**. **Kernel Space Device Driver** plays an important role by handling CPU and Memory Virtualiztion. However, **User Space Emulator** deals with the PC hardware emulation by simple using ioctl() system calls which is derived from QEMU[84].

2.9 QEMU

QEMU is basically a fast and well developed functional system emulation tool which is used in virtualization and Linux process simulation [13, 19]. QEMU emulator is free and open source and it can be used as a hypervisor like XEN and KVM [21].QEMU performs Hardware Virtualization and it uses Dynamic Binary Translation Technique which helps to achive a better performance(DBT)[19].QEMU emulator is very powerful and flexible to run unmodified guest operating systems and application programs. QEMU emulator along with KVM hypervisor can run several virtual machine and support architectures like X86,ARM and SPARC[28].

2.10 CPU Schedulers

The concept of CPU scheduling is when the CPU remains idle then allow one process from the ready queue to use the CPU and hold the execution of other process for quite a while because of lacking resources[22].CPU schedulers help multiple users to share the resources effectively and achieve Quality of Service(QoS). CPU schedulers have many criteria for example measuring Throughput, CPU utilization ,Turn around time,Waiting time,Response time and Load Average[23]. The main key role played by short time scheduler which picks up a process from the ready queue and send it to CPU.The scheduler works by maintaining two algorithms one is preemptive and another is non-preemptive.Preemptive scheduler selected a process which has higher priority for instance a process has priority 5 and it is running,then if another process comes up with priority 10 then CPU stops the current process deliberately and choose the next process until it will be finished.However,Non Preemptive scheduler is the reverse of preemptive scheduling algorithm if a process has given to the CPU it is not intervened or taken away until it finishes.

[20]https://www.linux-kvm.org/page/Main$_{page}$
[21]https://wiki.archlinux.org/index.php/QEMU
[22]http://www.studytonight.com/operating-system/cpu-scheduling
[23]http://www.studytonight.com/operating-system/cpu-scheduling

2.10.1 Xen CPU-Schedulers

Xen hypervisor also act as a hypervisor schedular which helps various virtual CPU to take their scheduling decision[14].The main functionality of the hypervisor scheduler is to decide which vCPU of the virtual machine will get priority to execute it's operation on host's physical CPU or pCPU[14].The most notable schedulers of Xen hypervisor are *Credit scheduler,RT-Xen scheduler*, *Simple Earliest Deadline First or SEDF scheduler* and *Borrowed Virtual Time scheduler or BVT schedular* [14].

- **Credit Scheduler:** Credit was the most popular virtual CPU scheduler which was derived from XEN hypervisor scheduler.Credit was the first among all the schedulers' of XEN hypervisor project and it is most notable to conserve the Symmetric Multiprocessing host or SMP hosts.Credit scheduler has also assigned a weight and a 0 cap which means it can able to run in any amount of CPU time[15]. The credit hypervisor scheduler has also some basic features like it is quantum based so it should have longer time slice that is 30 ms which denotes it can run 30ms before it can interrupt by other vCPU.Another one is mentioned before that it has 0 cap feature which means it can able to put Virtual Machines in work conserving mode.

- **RT-Xen Scheduler:** Real Time Scheduler which is known as RT-Xen is second most popular and open source virtual machine monitor which can support Xen in real time scheduling[16]. RT-Xen is the first real time CPU scheduler and widely used by the researchers and integrators due to it's attractive platform and provision real scheduling to guest Linux operating system[83].RT-Xen virtual machine monitor support embedded systems by provision real-time performance to the virtual machine images which will make sure that integrated subsystems be capable to fulfil their requirements[83].The main benefit of that particular scheduler is that it based on real time scheduling algorithm which makes it efficient,flexible and effective.It also set fixed-priority scheduling process in virtualized platforms.

- **Simple Earliest Deadline First(SEDF):** Simple Earliest Deadline First Scheduler also known as SEDF scheduler is also a weighted CPU scheduler which is based on real time algorithms like RT Xen and use time slice[14].The real time scheduling algorithm of SEDF was the modified version of Earliest Deadline First(EDF) algorithm. The SEDF scheduler consists of domains,each domain has it's own processing resources which also contains two parameters *period* and *slice* [56].The two variables has been created by two different purposes *slice* ensure the execution of the domains in a certain amount of time which is given by the *period* parameter[56]. SEDF schedular has the capability to work both conserving and non conserving environment.

- **Borrowed Virtual Time(BVT):** Borrowed Virtual Time or BVT schedular is the least well known Xen CPU scheduler which is ba-

sically a fair share and general purpose scheduler. BVT also contains domains which shares CPU time like SEDF scheduler[14].Burrowed Virtual Time scheduler is also quantum based and also have time slice but they are known as different name like *context switch allowance* and it also has symmetric multiprocessing capability but conserving work mode is disabled.

2.10.2 CFS Scheduler:

Completely Fair scheduler known as CFS scheduler is a process scheduler developed by Ingo Molner in October 2007[53].Completely Fair Scheduler has developed in Kernel version 2.6.23 which was basically the updated version of O(1) scheduler[81].The main purpose of CFS scheduler to allocate CPU resources among the executable tasks and ensure better interactive performance.The scheduler is mainly used the algorithm red black tree which is counting timeslice of the tasks just before it is scheduled for execution[44].Furthermore, CFS scheduler is exceptional than the other scheduler because of it distribute the CPU power equally among the tasks for instance in a multitasking procedure M process are is running simultaneously then every process will get 1/M seconds each for execution[44].

2.11 Control Theory:

Control Theory is a notable and well known branch in the field of Engineering and mathematics which dynamically adjust the systems behaviour.The mechanism is that it adds new element in the input chain of the systems that modifies the behaviours of one or more elements and modify the output chain [1] [54].Karl Astorm who is known as one of the top most inventor of control theory stated that control theory is a *magic of feedback* because it modifies the components of a system dramatically which performs lot better than the previous[65].A *controller* has to be designed on the basis of control theory which can control a system this term known as *plant*[54].The modified output signal known as *reference*.The *error signal* which is basically the difference between the modified output and the previous output has applied in the inputs of the system to get the desired control signal.

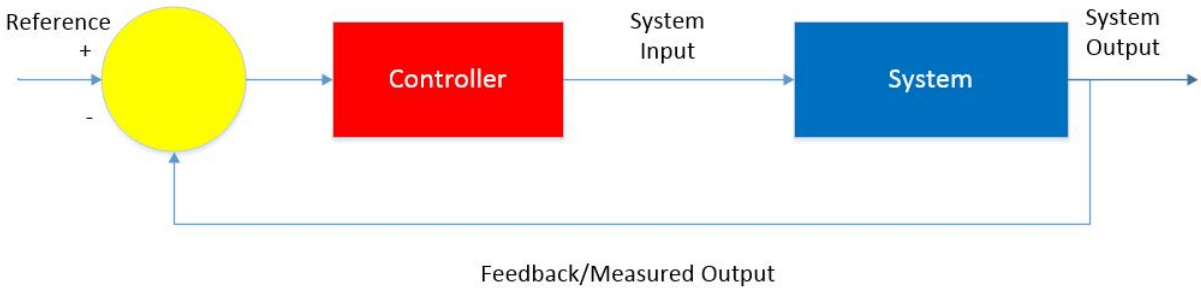

Figure 2.3: Diagram of Control Theory

2.12 Scalability-Horizontal and Vertical Scaling:

Scalability is defined as the capability of a system,network or process to handle the growing amount of work by increasing extra nodes or extra resource.Furthermore,scalability means to tackle the workload, by adding the resources, it's can be done in two ways by adding an extra server or hardware with an existing systems or add memory or CPU with the existing system. Scalability are of two kinds, Horizontal Scaling and Vertical Scaling.They are describing briefly in following.

2.12.1 Horizontal Scaling:

Horizontal scaling is to add or remove more computing resources for instance computer, server or distributed environment for the existing software systems[49]. Horizontal scaling is often known as scale in/out.The main advantage of horizontal scaling is that it can dynamically provision it's resources when it necessary.Horizontal scaling id more reliable and give better performance than the vertical scaling[3].

2.12.2 Vertical Scaling:

Vertical scaling is to add or remove computer nodes for instance memory,CPU or storage of a single node or single computing resources.Vertical scaling is often called as scale up/down approach.The main benefit of vertical scaling is that it can re-size without changing the code.Vertical scaling has less overhead and all the data is put on the single machine[52].Moreover,in vertical scaling managing data is easier than the horizontal scaling.The major problem of vertical scaling is cost efficiency,the cost is more when additional RAM or CPU set up with the existing machine than set up smaller instances[52].

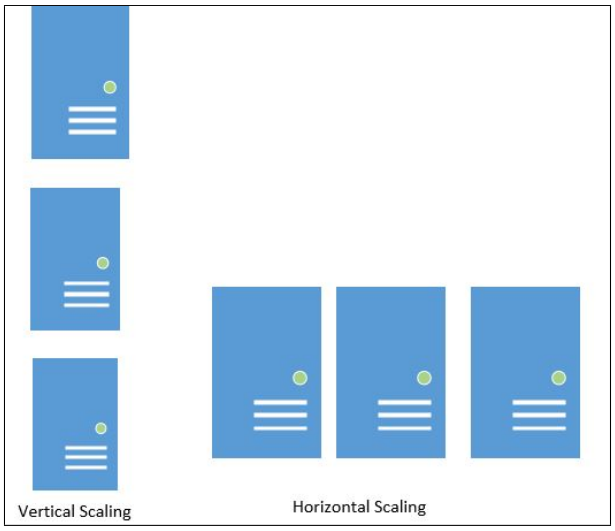

Figure 2.4: Diagram of Vertical and Horizontal Scaling

2.13 HttpMon:

HTTPMon is basically a monitoring tool that using HTTP protocol.The main task of HTTPMon is to check the status about the website by sending valid requests within a regular intervals.The benfit of the HTTPMon is that it can keep tracks of the uptime and downtime of a particular website.Therefore,large companies or organizations whose website has to be used in selling or buying staffs they are hugely benefitted by HTTPMon to check their website uptime performance and make the right decision regarding hostime,infrastructure and software that has been used by the website[57].

2.14 Libvirt

Libvirt is an open source application programming interface(API) developed by Red Hat [24].Libvirt is a widely used API to support orchestration layer of hypervisors.Hypervisors like KVM ,QEMU ,XEN, VirtualBox ,Microsoft Hyper-V, VMWare ESX and SGX, IBM PowerVM, Open VZ , Virtuzzo ,Bhyve are fully supported by Libvirt [25].Basically libvirt designed for c programming language.However to support other programming languages like for instance Python,Java,Ruby,Perl,JavaScript and so on libvirt has its own package named libvirtmod.The API also support local access control using Policykit. API also provide portable clients for Windows,Solaris and Linux.Libvirt also provide CIM for Distributed Management Task Force(DMTF) schema of virtualization. Libvirt also provide technologies to manage virtual machines, virtual networks and

[24]https://en.wikipedia.org/wiki/Libvirt
[25]http://libvirt.org/

storage.Libvirt also help to fetch information like VMâs status,CPU usage rate,VCPU number,virtual memory etc [34].

2.15 PID Controller

Proportional-Integral-Differentiator controller also know as PID controller is control loop feedback mechanism controller [58].The controller is the most widely used process control technique for many years [12].The first idea of PID controller has been given by a British scientist named James Clerk Maxwell [58].However, Russian American engineer Nikolas Minorsky developed the idea and bring it to the reality by using theoretical analysis and named it as PID controller [58].PID controller has several design techniques but most notable design has been made of by two American scientist named John G. Ziegler and Nathaniel B. Nichols which is also known as Ziegler-Nichols Method [12]. The most notable design method of PID controller consists of time-domain or frequency-domain performance criterion [12].PID controller basically calculates an error value which is basically the difference between set value and a measured process value. Afterwards, it applied three basic coefficients proportional, integral and derivative [60].

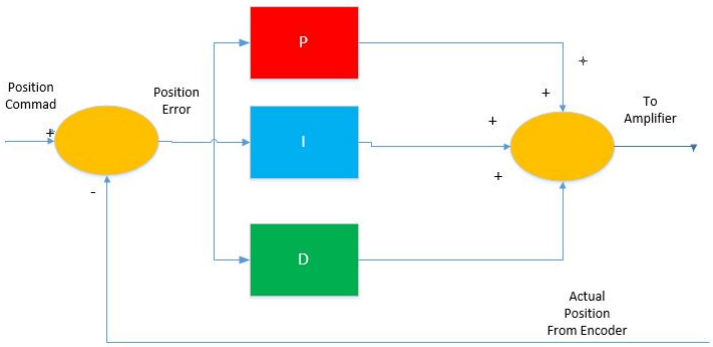

Figure 2.5: PID Controller

2.16 Related Work

Although the concept is pretty new but some notable research takes place in this particular field.Some of the research work has been discussed in this section.

2.16.1 Orchestrating Resource Allocation for Interactive vs. Batch Services using a Hybrid Controller

The research paper proposed a hybrid controller model which is based on control theory that co-ordinates real time and non real time applications[3].The hybrid controller is autonomic and increase server utilization

as well as the revealed resources guaranteed the Quality of Services(QoS). The main focus of the research is to maximize the server utilization within a limited amount of power consumption.Vertical and horizontal elasticity concept has discussed for better resource allocation.

Researchers study about different control models and their specific requirements.Afterwards They try to relate control theory with the controller models.The controller model has to be special feature like adaptive,scalable,rapid,reliable and robust[3].Controller Matrix deals with SLA parameters such as response time and frames per second.

In the experiment section the researcher has used 2 Dell PowerEdge R610 as physical machine.Xen hypervisor used as a hypervisor, for hyperthreading purpose 16 vcpu has been used as well.Moreover, 10.0.0.0/24 considered as public network and IPTABLES rules are also implemented.A bridge vibr0 was created for proper connectivity between VM'S and Internet. Apache open source server is used for web service besides apache various open source tools for instance Git,HAproxy,HandbrakeCLI,Libvirt,RuBBos and Loader. .

2.16.2 A Hybrid Cloud Controller for Vertical Memory Elasticity

The researcher proposed a synthesize controller which is designed with the basis of control theory named *hybrid memory controller*.The controller has the capability to scale up the resources if the system needs more resources or scale down the resources to avoid the wastage of extra resources[24].The main contribution of the paper is that the proposed controller adjust the size of RAM of the VM's by considering both memory utilization and application performance[24].The one of main creation of this paper is build up a new idea of vertical elasticity approach that is *Hybrid elasticity approach* which is basically the combination of Capacity-based approach and Performance-based approach.

They defined a new parameter named *control knob* parameter which solely took the decision about the scale up and scale down of the allocated memory.The decision has been taken on the basis of the response time(RT) and VM's memory utilization.The hybrid memory controller has 3 parts Controller,Sensor and Actuator.Sensors gather major real time information like *memory utilization* and *mean response time* which contributes in decision making.Controller is doing the major part of the entire system it dynamically tune the amount of memory according to the information provided by the sensor.Actuator contributes to increase or decrease the memory in each interval.

2.16.3 Autonomic Resource Provisioning for Cloud-Based Software

This paper comes up with an elasticity controller named RobusT2Scale which is created by using fuzzy logic.The controller automatically add or remove the resources according to the defined elasticity policy.Elasticity policy has been made by using type 2 fuzzy set[41].Threshold value was set for the autonomic controller,the decision such as CPU utilization has been made off by this particular threshold value.Elasticity rules has been applied depends on this particular threshold value,threshold values consists of only two variables *high* and *low*.

2.16.4 Coordinating CPU and Memory Elasticity Controllers to Meet Service Response Time Constraints

Farokhi et al. proposed an autonomic resource controller which has the capability to adjust CPU and memory requirements for the cloud based interactive applications[25].Resource controller was implemented in the basis of Fuzzy Logic.This paper researchers used three different benchmark tools like RUBiS,RUBBoS and Olio to estimate the workload on open and closed system models[25].The controller consists of four parts *fuzzy controller, cpu controller, memory controller* and *sensor*.*Fuzzy Controller* implements fuzzy logic in the system and it generates a coefficient in between -1 to 1, if it generate -1 then it means the resources are over provisioned and if generates 1 then it means resources are under provisioned.Fuzzy Logic Systems which is known as FLS has developed by the researchers to implement the fuzzy controller.Fuzzy Logic systems derived from the fuzzy rules and membership functions.(MFs)[25].They set some variables to design fuzzy logic systems,average Response Time(RT), average CPU utilization(U_{cpu}) and average Memory Utilization(U_{mem}) are set as input variable.On the other hand CPU coefficient(C_{cpu}) and Memory coefficient(C_{mem}) has set as output variable.*CPU controller* works as an adaptive controller,the job of the *CPU controller* is to adjust CPU cores for the applications by getting the value from the fuzzy controller.The value is measured by considering Response Time(RT) and the CPU coefficient(C_{cpu}).'*Memory Controller* has the same function like *CPU Controller* but the basic difference is instead of adjusting cpu cores it deals with the memory.It dynamically adjust the memory for the applications by the measured value of Response Time(RT) and Memory coefficient(C_{mem}). *Sensor* is the last part of the controller, the main task of *Sensor* is to allocate VM periodically by gathering information about mean response time,average CPU utilization and average memory utilization.

2.16.5 A Virtual Machine Re-packing Approach to the Horizontal vs. Vertical Elasticity Trade-off for Cloud Autoscaling

Sedaghat et al. proposed automated scaling system which is basically follow both vertical or horizontal scaling approach to handle the work-

load.The proposed autonomous system has the capability to dynamically handle the existing resources in both pattern either scale out or scale up approach [67].The system is cost effective and perform quite well.

Chapter 3

Approach

The chapter will outline a detail overview that how this research will go in progress.The main motivation of this research is that the experiment will be conducted based on the research questions and describe the questions in details.

1. *How can we determine a threshold value known as set value?*
2. *How can we build a dynamic PID controller considering the differences between the set point and measured value, and apply the correlation based on proportional, integral, and derivative terms?*
3. *How can we design and build a sophisticated Service Level Agreement (SLA) driven infrastructure to provide efficient QoS by applying better resource allocation?*

The research questions set a clear view how the paper will be in progress.The set value is played the most valuable part of this entire part of the research. The first phase of the research is to analysis the set value which will be the main part to implement the PID controller. Depending on the set value the PID controller will do the job.PID controller will increase the server utilization in order to use the concept of control theory. The fundamental logic of PID controller is it based on the mathematical logic of proportional,Integral and derivative and generates a value which will make the existing anatomic controller more dynamic and more reliable. The whole infrastructure should ensure quality of services(QoS) for interactive and non interactive applications.Moreover, Service level agreement is another key issue which should be maintained by the whole infrastructure.

The approach part will consists of following steps:

- Determine the set value.
- Design of the PID Controller Model
- Implementation of the model
- Experiments in different environment

- Data Collection
- Data Representation
- Fetch the resulting Value
- Analysis the expected result

3.1 Objective:

The main objective of this research was highlighted in the problem statement which consists of three questions that need to be solved precisely to figure out the whole problem statement. The terms and concept which is placed in the background chapter give a clear indication about the work and also provide some basic ideas of control theory and PID controller.

The main motivation of the research is to build an experimental scenario for PID controller logic which will enhance the performance of the autonomic controller to increase the server utilization. The set point is one of the main feature to build PID controller logic. The existing system already have installed server and the batch processing features are installed in the server system. Moreover, PID controller logic will perform the experiments in both vertical and horizontal scaling procedure to ensure better resource allocation. However, the existing autonomic controller used application level metrics for interactive and non interactive applications. Service Level agreement should be full filled to improve the performance the autonomic controller.

Overall the design pattern of this entire research will consists of three basic parts. Design part gives a clear view to the reader that how this research is conducted.The three basic parts highlighted in following.

1. Design Phase
    - Fix Set Value
    - PID Controller Model
    - Decision Model
    - Expected Result

2. Implementation Phase
    - Experimental Setup
    - PID controller logic implementation
    - Pattern the workloads
    - Conduct the Experiments
    - Testing Phase
    - Results from the experiment

3. Analysis and Result Phase

    - Collecting Data
    - Plotting Graphs
    - Analyzing the Graphs

3.2 Set Value, Process Value and Error Value:

The set value is known as the set point which will play the crucial role in the whole part of this research. Set value is also known as Control Value [66]. The set point is a non-negative weighted value that is basically a threshold value in this research. Set value or set point is the targeted value or desired value that needs to be achieved. Therefore, the value is known to describe the overall norm of the entire system [69]. The set value will be set according to the value of the desired response time.Ahmed and et al. [3] estimated the value will be the range of minimum and maximum average response time for instance if the average response time in between 100 and 500 ms. then we have to take the set value within range of 100 to 500. The set value should be more precise to count the error value. If the measured value is less than 100 then the value is ignored and instantly reduce the resources. Response time time is it in between 100 to 500 will be counted as normal and controller does not have to perform any operations . However. if the response time is more than 500 than the difference will be calculated between average response time and measured process value.

Process Value is known as measured value [66]. The measured value is the response time that is generated by the system. Moreover, the thesis work is to minimize the process value through the controller. The goal of the controller is to minimize the error value through it's mechanism. The error value is denoted by *e(t)*. Error value has been changed through the controller and generates a moderate value near to the set point known as control variable and it denotes as *u(t)*.

3.3 Features of PID Controller

PID controller is build on a topology based on proportional,integral and derivative terminology. The controller build in a control loop feedback mechanism. The most basic features of the PID controller it estimated the error value *e(t)* on the basis of the difference between set value and the measured value. Afterwards, it applies the topology of proportion, integral and derivative terms and make the correction of the error value and generate the controller output [58]. The controller sends the output to the existing autonomic controller to adjust the resources. PID controller itself is divided by three control based logic units and the logic units itself acts as a controlling unit Proportional controller or P controller, Integral controller or The controllers are describe briefly in following. The error value is the difference between the process value and the set value.

3.3.1 P Controller:

Proportional controller which is known as P controller. The goal of the controller is to generate an output value which is proportional to the error value *e(t)*. Controller estimated output value by multiplying the error value with a certain constant value [2]. This certain value is known as *proportional coefficient* and is denoted by the symbol *Kp*. The proportional coefficient is a non negative weighted value and it is determined when the controller is tuned [66]. The controller has some limitations for instance it never reaches im steady state condition. As a result steady state error is quite familiar in this controller. However, if the error is small then the P-controller could not be able to provide a better solution [66].

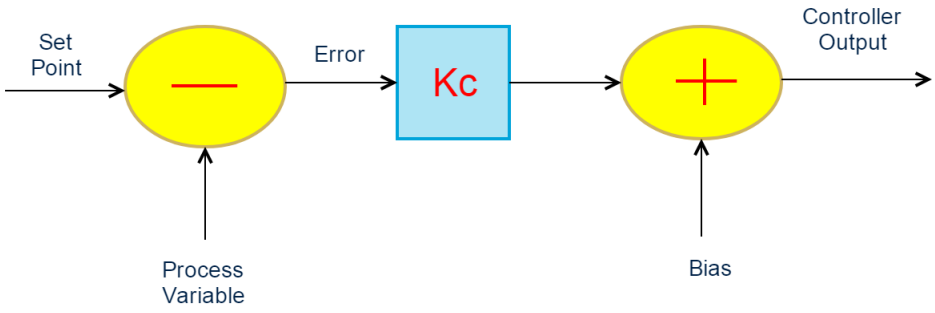

Figure 3.1: Diagram of P Controller

3.3.2 I Controller:

Another component of PID controller is I controller and I is originated from the mathematical term *Integral*. The integral controller comes to the scenario to remove the steady error. Major functionality of integral controller is to integrate the error value over a period of time until the error value has reached to zero [2]. After the integration part a resultant value has been generated which is called accumulated error value. In the next phase the accumulated error value is multiplied with a Integral Constant *Ki*. Integral controller has the capability to minimize the steady error and if the steady error decreases then steady error also decrease.

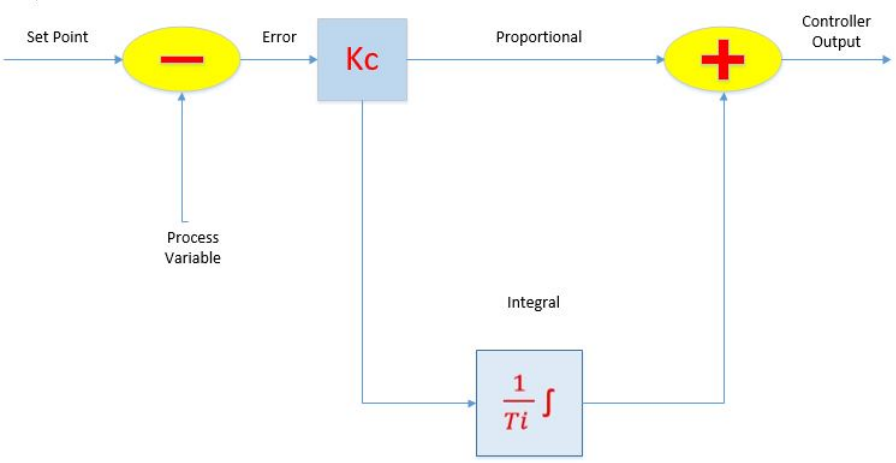

Figure 3.2: Diagram of PI Controller

3.3.3 D Controller:

The last component is the D controller like Integral controller it also originated from the mathematical concept *Derivative*. The rest of the controller for instance P and I controller does not have the capability to predict the future behaviour of the error but the D controller is an exception. Derivative component of the controller is more necessary to compensate for sudden changes in the error [66].

3.3.4 Combination of the controllers:

Though the three main components composed the actual PID, but most often the P and I controller has been used because of the D controller or the derivative controller has the lack of finesse [74]. Moreover, the D controller is difficult to implement. In spite of the problems of D controller, D controller is anticipating the future behaviour of the error [2]. P, I and D individual controller has several drawbacks, also integral and derivative action to a proportional-only controller do not generate the appropriate outcome [74]. Moreover, the I controller is too much aggressive which makes PI controller is bit complex. So, the overall combination of all these three functions make the controller quite stable and robust which can generate an appropriate control signal to get the desired response from the controller [66].

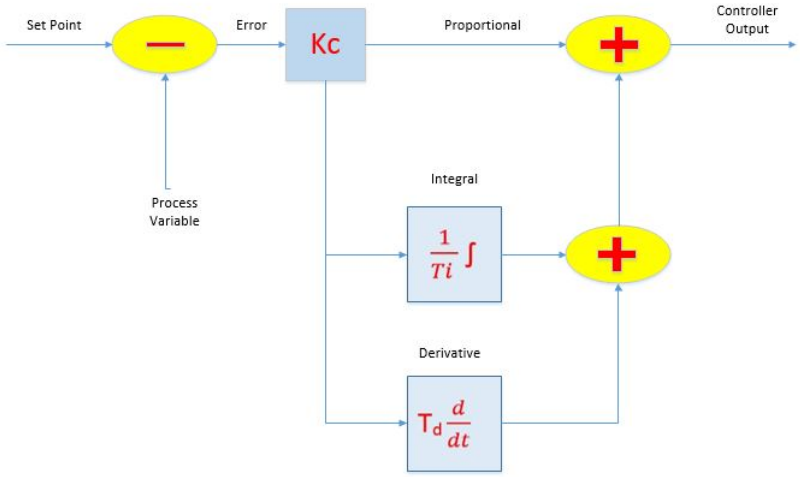

Figure 3.3: Diagram of PID Controller

3.4 Algorithm of PID Controller:

PID controller is a combination of mathematical function like proportion, integral and derivative. It follows a mathematical equation. Moreover, the behaviour of the controller maintains control theory mechanism which has been stated on briefly in the background chapter. Controller has generated a process value *u(t)* through a mathematical function and minimized the error value. PID controller has followed a simple algorithm to generate the above mentioned process value although there has been many algorithm has been followed by the PID controller. However, in this research a simple and most well known algorithm has been discussed.

PID controller has several well known algorithms but most common algorithm is the Zieglers Nichols Algorithm. This study will brief discuss how this algorithm will work and contribute this research.

$$CO = K_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}][75][4] \tag{3.1}$$

e(t)=Error Value
Kp= Proportional constant
CO = controller Output
Td= Derivative time constant
Ti= Integral time constant
Kp = Proportional Gain

$$CO = K_p e(t) + K_i\int_0^t e(\tau)d(\tau) + K_d\frac{de(t)}{dt}[75] \tag{3.2}$$

The first equation resembles the mathematical function of getting the control variable. Td and Ti are the time constant for derivative and integral

controller consecutively. Whenever Kp is multiplied by the Integral time constant we get a new constant that is called Integral tuning constant or *Ki*. Similarly, whenever Kp is multiplied by the derivative time constant new derivative tuning constant has been generated which is expressed by *Kd*. However, in our PID controller we are directly work with the tuning constant parameters such as *Ki* and *Kd* instead of using the derivative time constant *Td* and integral time constant *Ti*. The parameters *Kp* , *Kd* and *Ki* should be estimated optimally so that the closed loop system has to give desired response [6].

3.5 Autonomic Controller Model:

The autonomic controller has designed by the researcher *Bilal* for decision making purposes [3]. Based on the decision it perform both horizontal and vertical scaling by allocating resources. The control theory is used for decision making. Autonomic controller model is basically adopted the model of hybrid controller model and like other controller it is base on some parameters. Two parameters rt_k and rt_i is defined, the the desired value is set as rt_k and rt_i set as measured value. An *error* value is set which indicates the difference between the two values. Parameters $Umem_i$ and $Ucpu_i$ deals with the utilization of memory and cpu respectively. In addition controller allocate CPU and memory by two parameters cpu_i and mem_i.

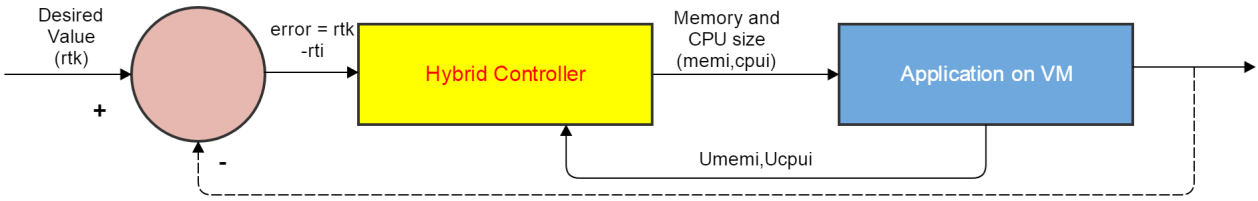

Figure 3.4: Autonomic Controller

3.6 Design Phase

PID controller is more popular and widely used controller in the industry because of it's simplicity and robust performance. Though PID controller is much effective controller logic and provide good solutions for some control problems but in some cases it cannot provide optimal control [58]. Moreover, small number of parameters are used for tuning the control mechanism also tuning methods may not provide satisfactory closed loop responses in some circumstances [5]. We have to consider all those issues

to design the controller and make as much efficient to provide better and fair result from the controller. PID controller logic has been implemented with the existing autonomic controller. Both PID and autonomic controller is based on the control theory and PID is follow the close loop feedback mechanism. Decision made by PID controller and the task of autonomic controller is to the resources for instance by doing vertical and horizontal scaling.

3.6.1 Decision Model:

PID controller is a feedback model controller that maintains a sudden algorithm. The prototype of the PID controller maintains a certain algorithm to generate the control output. A build in sensor lies on the prototype which has the ability to compare the generated response time which is identified by the process value and compare with the maximum response. The whole operation has been divided in two phases .The first phase is handled by the prototype of PID controller and the second phase has been done by the existing autonomic controller. The existing autonomic controller consists of four sub models which has the ability to work simultaneously [3]. Autonomic controller is also based on some parameters which contributes to perform Vertical and horizontal scaling [3]. We are going to give overall descriptions of the whole mechanism in the implement section but in this section we just highlight every phases in following:

First Phase: PID controller is going to perform the operation in the initial stage.

- Generate the process value by the sensor.
- Based on the process value or PV an error value has been estimated.
- Error value has gone through the process of PID.
- The output signal has been generated through the PID.

Second Phase:

- Collect the information from the PID
- Decision to perform Vertical Scaling
- Decision to perform Horizontal Scaling
- Confirmation of the final state.

Whenever the first phase has finished the operation then the second phase will start. In the second phase the vertical scaling and horizontal scaling both are going to add new resources. However, in vertical scaling a new CPU or memory is added and in horizontal scaling a completely new VM is added.

3.6.2 Design of Controller Metrics:

We need to perform precise actions to show the overall analysis. Metrics are the measurement of the controller behaviour which denotes how controller reacts and how it adjust resources for real time and non real time applications [3]. To perform correct actions metrics plays an important role and metrics will be collected from the VM's [3]. Webservers are always latency critical and for calculating the response time for real time applications we have to measure each state of the webserver. Non real time applications for instance the batch processing the state will also checked like real time applications. Though the research will focus much less in non real time applications. The metrics which will consider in this research will be stated in following:

- Response time of Webserver
- Frame Per Second or FPS of Video encoding
- Increase or decrease CPU cores
- Allocated or dis-allocated of RAM

Response time of the webserver is defined by as total amount of time is needed to respond a particular request. Large number of request is sending each seconds in the webservers and response of those request has been generated every moment. In this thesis we are going to generate request of 250 or 500 clients and calculate response time in milliseconds or ms. However, in the case of batch processing software consumes more CPU power to do video encoding. Therefore, to ensure QoS we have to measure the non interactive batch processing applications by Frames per Second(FPS)

The whole part of of design of the controller metrics has done already in the autonomic controller in the previous thesis. Therefore, I have given some basic ideas of the design of the controller metrics. Control metric has designed by Bilal and et al. in the existing autonomic controller.

3.7 Stages of Implementation:

The implementation procedure consists of several tasks that needed to be completed before going to the final experiment. Moreover,in implementation stage we need to setup an outline for building up the experimental stage which consists of several important tasks. The tasks will be discussed in following:

- Experimental Setup
- Implement the controller Logic
- Generate Workloads

- Necessary Tools
- Define Metrics for Interactive and Non-Interactive Applications
- Conducting Initial Experiment

3.7.1 Experimental Setup:

Experimental setup gives a clear vision of our entire implementation procedure. After designing the PID controller logic we need to configure the physical servers. The physical servers are setup in Linux environment of Ubuntu 12.0.5. Two physical servers are build up with same platform for this entire research project. Physical servers should be reliable in terms of performance and security.

To build up virtual environment we need to install hypervisor and VM's. Therefore, we have install Xen hypervisor which is a type 1 bare metal hypervisor and the installation procedure of Xen is simple than other hypervisors for instance KVM and VMWare. However, in the case of XEN hypervisor installation disk partition has to be performed during the time of installing the operating systems [3]. A dynamic disk partition tool named Logical Volume Manager or LVM has been set up for partitioning disks. LVM is a common and most well known tool that has been used more often for disk partitioning and LVM has the capability to manage large disks by allowing disks to be added and replaced by without downtime or service disruption [47].

In the final phase of experimental section we have to configure the networking and allocate separate subnets for the VM's. Moreover, an established connection between the servers is set up to ensure high performance of each servers [3]. The servers has the ability to host three different kinds of VM's. The are listed in following:

- Webservers
- Batch
- Database

Implementation of Database and Webserver we have followed 3 tier application systems [33]. Webservers deal with Presentation layer and Business/Domain layer, whereas Database handles data storage layer. Batch is dealing with the batch processing jobs for instance video encoding. The SSH or secure shell connection will be establish for security and authentication purposes.

3.7.2 Implement the Controller Logic

The prototype of the controller logic has to be implemented with the existing autonomic controller. The prototype of the pid controller logic

and the existing autonomic controller both are derived from the control theory. PID controller logic has been set up with the existing autonomic controller to make it more efficient and robust. Moreover, the foundation of the whole research lies on the PID controller logic implementation which can contribute to achieve the desired level of QoS for server utilization [3].

PID controller logic is the main foundation of this particular research. The controller is based on a threshold value which has the capability to control the resources or allocate the resources in proper way. PID controller operation is so simple than other controllers [39]. The logic of PID is easy to implement, but the parameters has to be set carefully. Moreover, implementation of PID depends more in the parameters value. The parameter set value, Kp, Kc and Ki will be set to run the PID. In our case we write a python script to configure the PID. Also we have tested with different set value to test the PID controller. Finally, run the code to execute the PID.

The existing autonomic controller is also written in python script [3]. Like PID controller the autonomic controller has the potential to make decisions. Autonomic controller has the capability to perform vertical and horizontal scaling. Furthermore, controller has the ability to collect the metrics from the running VM's [3].

The major thesis work is to relate the PID controller with the existing autonomic controller. PID controller has the capability to send the signal to the autonomic controller. The major work is to make the decision precisely and appropriately. PID take the decision based on the response time for the webservers and frames per second for the batch processing files. Whenever PID take the decision it generates the control signal which sends to the autonomic controller. Afterwards, autonomic controller provide the resources by performing scaling. Vertical scaling has performed before horizontal scaling. CPU and memory has been provide based on the availability of the resources, whenever the resource allocation is completely done then it performs the horizontal scaling by providing a new webserver.

3.7.3 Generate Workloads:

The workloads has to be measured in several ways. Many tools have been used so fa too measure the workload for instance Loader.io , Autobench and Httpmon. In this case we have generated workload through *Httpmon*. *Httpmon* is a simple monitoring tool that has been design to monitor the response time. The installation procedure is simple than other tools and it generated the workload and save it in a log file.

The two physical server we are dealing with in this research. One is named *bilal1* and another one is *bilal2*. The installation procedure has taken place in *bilal1*. Before installing *Httpmon* we have to install a revision control system. *Git* is most well known and lightly used revision control system. After installing *Git* in our physical server we have to configure the *Httpmon*.

3.7.4 Necessary Tools:

To create and test the environment we need some tools to use. As mentioned before that generating loads we need *Httpmon* and *Git*. Besides this two tools we need more tools for building the environment of the entire systems. Most of the tools are open source and configured manually. Moreover, almost all of the tools are easy to install and user friendly. However, tools have the system dependencies for instance the physical servers are running on Ubuntu 12.04 which is bit old therefore we need to keep an eye to installing these tools. The tools are discussed in following:

- **Apache2:** Apache2 has been installed for running both of the webservers.
- **Autobench:** The benchmark tool has used to generate the average response time of the server by sending the amount of HTTP request and present it in the graphical manner.
- **Git:** Most used open source revision control system Git has been used in out experiment.
- **Httpmon:** Httpmon used as a monitoring tool basically used to survey the state of the running webservers by sending valid Http request. Httpmon used the HTTP protocol.
- **RuBBoS:** RUBBoS is particularly a bulletin board benchmark application and it was by JMOB [63]. It allows users to browse and post their comments and activities.
- **Python:** Python the scripting language we used for our tools.

3.7.5 Define Metrics for Interactive and Non-Interactive Applications:

The application level metrics is the measurement criteria of the webservers and batch processing software. The level of metrics predefined in our analysis terms. Our system needs to achieve a certain goal which particularly depends on the level of metrics. However, the criteria is differed for interactive and non interactive applications.

The *response time* is the measurable criteria for the interactive applications. The task of a *response time* is defined as the time elapsed between the time when a task is ready to execute to the time when it finishes its job [62]. Basically the response time is the amount of time is needed to finish a particular service for instance memory fetch, disk I/O , database query and loading website [62]. Several causes are responsible to generate high response time such as slow database query, high CPU usage, memory starvation and poor performance of server. Our goal is to maintain a desired level of *response time* within a certain interval to minimize the Service Level Agreement(SLA) violation [3].

In non interactive applications a software service has used so far to measure the *Frames per second*. The software is known as HandbrakeCLI. HandbrakeCLI basically a command line interface which enables video encoding, decoding and conversion [35].The average frame per second should be in a decent range for instance in between 15-20 not below than 15. Otherwise, it counts as SLA violation.

3.7.6 Initial Experiment:

The experiment will conduct with two different phases. Experiments will show how our prescribed system will work and help us to make final decision about our system. The first phase of experiments lies our observation about how changing CPU cores influence to map the VM's which is known as *CPU hotplug* Next phase we are performing the same operation instead of changing the CPU cores we will change the memory that is called *memory hotplug*.

3.7.7 CPU Hotplug:

Hotplug means adding or removing any components and add the capacity of the total system without switching off the whole system [37]. The *CPU hotplug* means the adding or removing the vcpu on a VM from the hypervisor without closing or rebooting the VM [3]. We need to do *CPU hotplug* because of testing purpose, we need to give more resources to perform that how long it takes to VM to add the vcpu by adding or removing them.

The hotplugging has taken place in the phase of vertical scaling. The whole experiment conducts in a two phase. First phase we increase the CPU one by one and measure the amount of time it takes to map the VM. Next phase we decrease the vcpu in the same rate and examine the time.

3.7.8 Memory Hotplug:

Memory hotplug has the same conception like CPU hotplug. In stead of adding CPU it is adding or removing memory from the system.

We will conduct memory hotplug in two phases, in first phase we will test with 8GB. Afterwards we will increase the memory and make it 10 GB and try to calculate the to map the VM. At the same we also decrease the memory from 10 GB to 8 GB and calculate the estimated time.

3.8 Experiment:

The section will focus the overall experiments and their prerequisites briefly. This phase will conduct after the completion of the initial experiments. Initial experiments provide some basic ideas to the reader how the main experiments will be conducted. In the phase of the experiment, we

also have conduct some of the proposed sample experiments which can generate the results and generate the data for the analysis stages.

The structure of the sample experiment stages contribute to build a proof of concept of the contribution of this particular research. The total structure of the sample experiment is stated in following:

- Control Interval
- Webserver (Vertical Scaling)
- Webserver (Horizontal Scaling)
- Batch Processing Experiment

3.8.1 Control Interval

In this experimental phase we have to test the prototype before the main experiment, therefore we have to find a stable interval to test our prescribed PID controller prototype [3]. We will generate a response time in two phases of interval for instance we will set a valid time duration for generating the response time. The time duration will be stated as following:

- 10 seconds
- 20 seconds

However, for the batch job to generate Frames per Second(FPS) takes longer time for instance the interval is around 10 minutes.

3.8.2 Webserver(Vertical Scaling):

In this section of the experiment performs with the real time latency critical applications and allocate nodes to perform vertical scaling procedure. Moreover, the workload has been generated accordingly. In this experiment we have to generate workloads in two different phases. Firstly generate the workload for 1000 active users and test the prototype can handle it effectively. Secondly the same experiment will run for the 2000 active users. If the response time is increased the controller will allocate the resources accordingly. On the other hand, if the response time is decreasing than the resources are reduced. However, when all the resources have been used then the goal of the latency critical application to *steal* all the resources from the batch and provide it accordingly to meet the desired level of throughput and avoid the SLA violation [3]. Furthermore, the system has the functionality to fully avoid the conflict of the resources.

The purpose of this experiment is to observe that the prototype has the capability to allocate the resources successfully for the latency critical applications and maintain the desired level of throughput and avoid the SLA violation and maintain the QoS. Moreover, the controller system has to be maintain a solid cooperation between real time and non real time

application to provide resources from the batch job to lend it from the batch and allocate it to the webserver. Whenever there is no shortage of the resources then it also give it back to the batch.

3.8.3 Webserver(Horizontal Scaling)

This section will discuss about a major issue suppose the workload increases randomly and it's duration stays for longer period also the batch is running out their resources then we have to perform horizontal scaling.To perform horizontal scaling we have to set up a new physical machine and bootup seconds new webserver. Also add up load balancing configuration to balance load between the two webservers.

In horizontal scaling we have to add a new webserver that will boot up and configured in the second physical machine. Load balncer has configured with the newly formed webserver. When the newly formed webserver is added to the system, the first webserver reduce half of it's resources immediately for the batch job. Second new webserver used half of it's resources together with the first webserver and tackle the workload as long as workload tends to become normal. Afterwards when the workloads going back to the normal stage, the system shut down the second webserver and reduces the resources. Therefore, the system goes back to it's previous stage and perform vertical scaling as before.

3.8.4 Batch Processing Experiment:

Batch processing jobs are only CPU intensive but it used minimal level of memory. The batch jobs are not latency critical applications. Batch processing files deals with the video encoding. We perform the batch process which estimated number of frames in one seconds then we show how much CPU it utilized. For this experiment we configure new VM called *batch* and perform operation onto it.

3.9 Data Collection and Plotting:

The PID controller maintain a desired level of response time per second(RPS) for the real time applications and frames per second(FPS) for the non real time application by providing the resources according to the defined policies. To proof that the system is functioning properly and do the operation for instance changing cpu and memory properly we need to generate the data precisely. Moreover, data reflects that the accuracy of the system and proof that our implementation of the whole system is working according to our prescribed terminology. However, for collecting data there should be certain prerequisites. Most desired prerequisites are the two physical servers, two webservers , database servers and batch processing service should tuning fine and accordingly.

Whenever the data has been collected then we have to represent it as a

graphical ways. Data has been collected the from the VM's simultaneously. Cpu script is running on the VM's which can give the actual usage of CPU. On the other hand the memory usage also measure with a bash script. After getting the data we have to plot line graphs to represent them.

3.10 Analysis:

The last phase of the approach part will consist of the most important section of analysis. The task of this phase is to analyze the plots and charts which are obtained by plotting data in the previous phase. Analysis phase will help us to find the ways how can it develop it in future which is regarded as future works and draw a fair conclusion.

The most crucial part of of analysis is to check the behaviour of the pid controller system, also observe that the PID controller can synchronize in a right manner with the existing autonomic controller. Also a major point of the consideration includes how the controlling system perform in the decision making phase and minimize the desired threshold value to allocate proper resources without SLA violation. Furthermore, to maintain the Quality of Service(QoS) the controlling system should truly determine the resources and avoid the misuse of resources for instance task is to avoid over or under provision of resources [3]. To avoid the violation in decision making the controller should have to consider the following [3].

- Response time < 500 miliseconds.
- Average Frames per Seconds > 15 frames per second.

Chapter 4

Design and Models

In this chapter the implementation of the controllers both the PID controller and existing autonomic controller will be described more than in the approach and the design part of the approach. Moreover we are going to focus deeply about our overall operation of the whole controlling systems which will reflect how can we adjust the resources. In the last phase of this chapter controller metrics will contains information on how we set the criteria of adjusting the resources which will help us to design an algorithm in the next chapter.

4.1 Controller Characteristics:

PID controller and autonomic controller both are derived from the control theory, also they are working as feedback closed loop controllers. As we describe the structure of both of the controllers in our approach chapter. Feedback controller has some potential anticipation than the open loop controllers. Feedback controllers have some basic characteristics [18].

- Steady State Accuracy: A process is called in steady state if the parameters which define the behavior of the system or the process are unchanging in time [70]. Whenever, a system comes to a steady state, the difference of input and output is measured [71]. To predict the accuracy in a steady state of a control system, a standard measure of performance is widely used which is known as Steady State error. Steady state error occurs when the the output of the system at steady-state does not exactly agree with the input [71].

- Stability: Stability of a controller indicates small changes in input for instance reference input etc.and any other initial conditions do not occur any large changes in the control system output [18].

- Disturbance Rejection : In real time control system it is always demanding to design a good controller which can generate desired control output because there is always some external noise that can change the actual output. The distribution and magnitude of the disturbance or noise relies on the working environment, and

sometimes it is too difficult to avoid the noise from happening [20]. However, PID controller has the ability to reject the disturbance by using the output to shape the input of the system [79].

- Sensitivity: In the structure of a feedback controller the parameters are matched to the system process. Sometime, the process variation can occur and change the parameters of the controller which can make the system unstable [68]. Therefore, the sensitivity is another key issue for the feedback controlling system. Parameters of the controller should have chosen carefully so that the process variation does not the effect the sensitivity characteristics of the controller [68].

- Bandwidth Extension: Bandwidth extension denotes to the deliberate process of expanding the frequency range of a bandwidth signal in which it contains an appreciable and useful content, and the frequency range in which its effects are not so small [7] .Control system has the capability to expand it's frequency range when it is necessary.

4.2 Controller Models:

The desired characteristics of the feedback controller are stated in section 4.1. The existing autonomic controller has also been designed based on on some ideal controller models [3]. Two types of controller models has been prescribed, one is *capacity* based and another is *performance* based controller. The main evaluation criteria of the controller is their behaviour and performance. The prototype of the of the autonomic controller has made based on the combination of these two models [3]. The controller models has been described briefly in below.

1. **Capacity-based Controller Model:** This model mostly based on the concept of allocating the resources properly and contribute the level of utilization [3]. Capacity based model is well known for it's simple architecture so that the model is widely used for performing vertical scaling [3]. However, there is some drawbacks in capacity based model for instance the utilization of resources of real time applications cannot assure the proper Quality of Services(QoS) which might be lead to over-provisioning of resources.

2. **Performance-based Controller Model:** In performance based controller model the level of resource utilization strongly emphasized on the Quality of Services(QoS) in real time applications. Many researches found out that the performance based controller model increase resource efficiency than the other controller models. Performance based model gather information from the application level metrics such as response time, and perform the decision based on the criteria. Application level metrics provide a clear indication about

latency critical applications. Controller itself has some parameters for instance acceptable and non acceptable values which largely contributes in decision making. However, like other controlling models performance based controller models has some drawbacks. One of the major limitation is that the performance based model decision making criteria mainly based on the application level metrics which does not indicate, which resource or resources are causing the degradation of Quality of Services (QoS). Moreover, the model also requires collection of resource utilization to perform decisions in a more efficient way to reduce resource wastage [3].

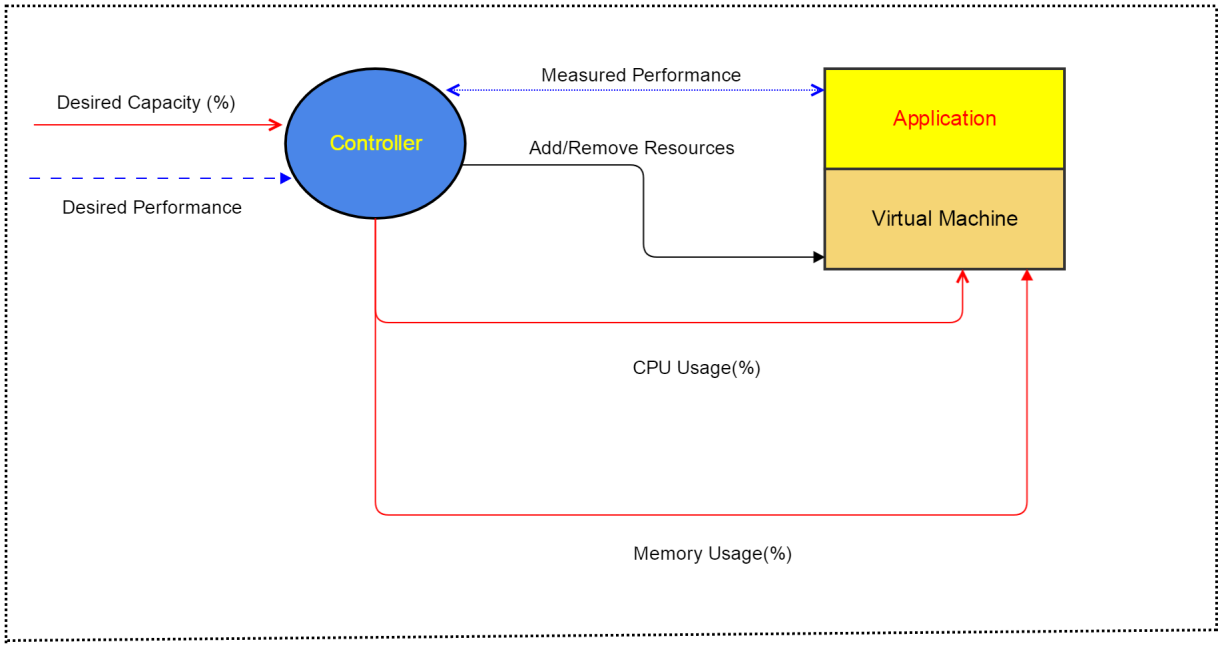

Figure 4.1: Capacity and Performance based Controller

4.3 Performing Operation:

The decision model was described briefly in section 3.5. We have just highlighted the basic principles of our overall decision model. PID controller handles the major decision logic about the whole system. As described in section 3.1 is that the set value plays a key role in decision making. Set value has been manually configured in the whole system. We will described more about set value in the last section of this chapter. In this section we will describe how our prescribed controlling system performing the mechanism in overall. The whole mechanism is divide in five phases. First three phase the PID controller perform the decision based on proportional, integral and derivative and generate the controller

output(CO) based on the process value. Also the controller generate a controller output and send it to the autonomic controller. In following the whole operations of the control system will describe successively.

Perform Decisions		
Phase Number	Controller Name	Perform Activity
1	PID Controller	Find out the Process Value(the measurement of QoS The average Response Time / Frames Per Second) and compare with the desired amount of QoS.
2	PID Controller	Determine the error value and apply correction based on Proportion, Integral and Derivative.
3	PID Controller	Determine the Controller Output and send back to the Autonomic Controller.
4	Autonomic Controller	Check the CPU and Memory Utilization.
5	Autonomic Controller	Perform actions if needed.

Table 4.1: Decision Performing

Set value is the measurement of desired level of QoS and it has been fixed manually by the system administrator . PID controller compare the set value with the generating process variable in sudden time interval. The controller has the capability to check the process value of the environment. A sensor has been built with the PID that can check out the process value. Time interval is also fixed before performing the operation. If the process value is equal to the set value then no operations will perform. Another most important task of PID is to maintain the desired level of QoS.

The autonomic provision controller task is to get the output from the PID and allocate or reduce memory or CPU. However, the controller can check frequently how much memory or CPU has been utilized by the system. The controller has two specific parameters $Ucpu_i$ and $Umem_i$. Parameters are the equivalent of CPU and memory utilization.

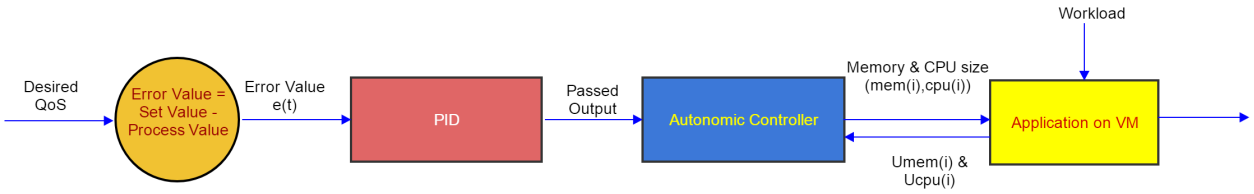

Figure 4.2: Design of overall Control Systems

4.4 Metrics of Controller:

The controller metrics has defined for webserver and batch jobs separately. The main contribution of this project is the set point which is particularly dealing with the SLA violation. Study shows that human brain cannot differentiate the changes which happens with 100 miliseconds []. Moreover, a human eye can detect changes when it is at least 1 seconds. Therefore based on this research we will keep our average response time within 100-500 miliseconds. Average response time more 500 miliseconds will be recorded as SLA violation [3]. However, if the response time will drop than 100 miliseconds which indicates the drop of workload, so forth the resource will reduce instantly. The extra usage of resources cause redundancy and QoS does not ensure properly. The following tables will give a clear view about the Service Level Agreement which we defined for the webserver.

SLA Violation	
Average Response Time	SLA Criteria
Fast	less than 100 miliseconds
Accurate	100-500 miliseconds
Slow	More than 500 miliseconds

Table 4.2: SLA Violation of webserver

In batch processing files SLA violation policy is bit different than the policy of webserver. The measurement criteria of batch processing files is Frames per Second or FPS. In this case the lower priority is counted as Service Level Agreement(SLA) violation. Average FPS should lies in between 15-20 and below 15 FPS has been counted as SLA violation.

SLA Violation	
Average Frames per Second	SLA Criteria
Fast	more than 20 FPS
Accurate	15-20 FPS
Slow	less than 15 FPS

Table 4.3: SLA Violation of Batch processing

Chapter 5

Implementation

In this chapter includes the whole experimental setup of our entire research project. Experimental setup consists of the network configuration, virtual machine setup, configuration of load balancing such as HAProxy , generation of workload through HTTPmon and Autobench. Moreover, an algorithm is also designed which will reflect how can we allocate the resources. The algorithm contains information on how the design became and how this was done. The algorithm will be represented as a flow chart which can clearly describe the logic of the whole system.

5.1 Experimental Setup:

The set up of network system consists of two Physical Machine(PM) of of DELL PowerEdge R610. The machine was set up in Oslo and Akershus University College [3]. Two machines were already setup for the previous experiments. To perform the experiment, we are allowed to get access to both of the machines and manage to control all the resources. Both of the machines have same specifications and running the same operating systems Ubuntu 12.04.5 LTS. Xen is the only bare metal hypervisor which supports all these specification and for it's simple architecture, we used it in our experiment. The following table stated the configuration of the Physical Machines.

PM 2xR610	
Components	Configuration
CPU	2xQuad-core Xeon E5530 2.40 GHz
Memory	24GB of 1066 MHz
Disk	2x146 GB
Network	8xEthernet Ports

Table 5.1: Physical Machine Configuration

To perform vertical scaling we need to increase our nodes for instance,

memory and CPU. Therefore, we have allocated used 16 CPU and 24 GB of Memory to do the vertical scaling procedure. Hyper threading is enabled in the CPU. The PM's are connected each other through an interface *eth1* and the bandwidth capacity is allocated about 1 Gbps [3]. Moreover, the Physical Machines(PM) have Internet access and secure shell access from outside.

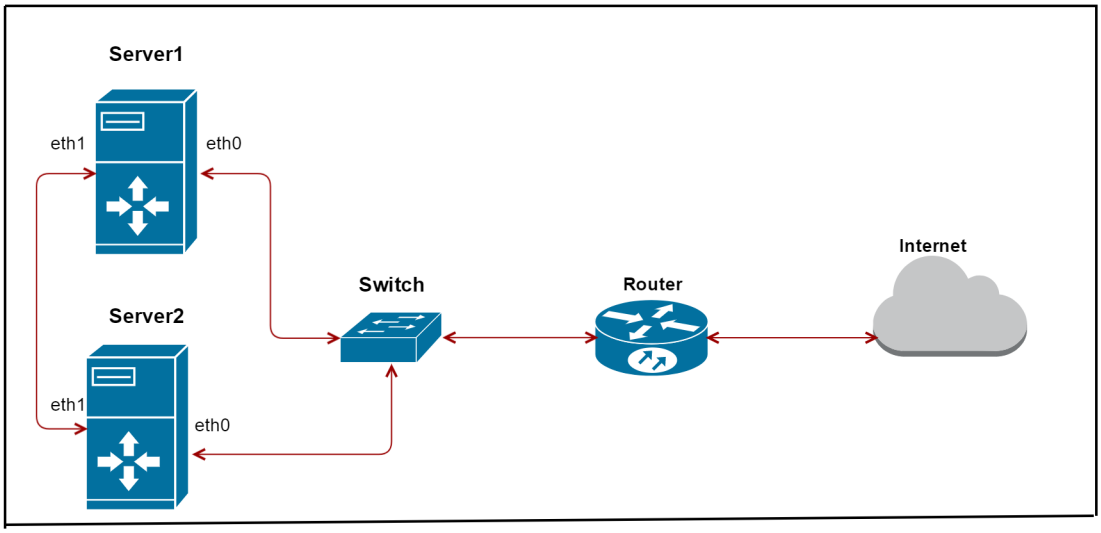

Figure 5.1: Network Setup

5.2 Network Setup:

The VM's which are created for the experimental setup were configured with a private network 10.0.0.0 and Subnet is 255.255.255.0. Server1 and Server2 has named *Bilal1* and *Bilal2* successively. The firewall rules of the two servers were configured to allow traffic outside from the main network. Moreover, NAT rules are also configured to allow internet access through whole network.

Network Overview Table			
Physical Machine	Interface	IP	Subnet
Server1	eth0	129.39.120.25	255.255.255.254
Server1	eth1	10.0.0.1	255.255.255.0
Server2	eth0	128.29.120.26	255.255.254.0
Server2	eth1	10.0.0.2	255.255.255.0

Table 5.2: Network Overview

Server1 and Server2 is directly connected with the Internet through the interface *eth0*. To ensure Internet connections from the other network peripherals we have to apply the IPTABLES rule for **POSTROUTING** as

well as **MASQUERADE**. The IPTABLES rules are stated in List 5.1. The bridge *vibr0* has been configured to ensure network connection between the VM's internet connection which is attached with the interface *eth1*. The other connections such as tcp and udp connections are **ACCEPT** in both ways from webserver, batch and database to servers or vice verse.

Listing 5.1: IPTABLES Rule

```
iptables -I FORWARD -i virbr0 -o eth0 -m state --state
NEW, RELATED, ESTABLISHED -j ACCEPT

iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
```

5.3 Virtual Machine:

As mentioned in section 2.7 that Xen has a special domain called *domain0* or *Dom0* which contains drivers for the hardware, as well as the toolstack to control VMs.[82]. Before installation the Xen hypervisor the Domain *Dom0* was installed. Besides,*Dom0* Xen hypervisor has other guest which also known as guest domain or *DomU* but the *Dom0* is the most privileged domain among them.

The main drawbacks of Virtual Machine(VM) is that it needs more storage. It can be solved in two ways either installing them from VM or creating the Logical Volume Manager(LVM) partition. In addition the LVM partition is bit risky and bit complex. Therefore, we have selected the disk image procedure which is more simple and convenient than the LVM partitioning. To perform the installation procedure first we have to create the disk image which later used to bootup the virtual machines for instance the *webserver*, *database* and *batch*.However the size of the disk image is 10 GB or 20 GB based on the purpose of the VM. The commands that are used to create the disk image are listing in list 5.2.

Listing 5.2: Disk Image

```
dd if=/dev/zero of=template.img bs=8388608k
seek=6144 count=0

mkfs -t ext3 XenGuest1.img
```

Disk image is stored *.img* file later it used fore creating the configuration file for the VM's. Each configuration files used separate disk images file for installation procedure. The configuration of the VM's stored in VM.cfg files which consists of the total amount of memory,number of vCPUs, IP-address, MAC-address, virtualization technique and so on. Each of the VM is managed through the VNC console and access inside the VM which can help us to do the troubleshooting for instance if any of the VM is crushed then we can fix it through the VNC. In addition to change the network

or Secure Shell(SSH) configuration we are to do it through VNC [3]. The Virtual Machine configuration file has been listed in list 5.3.

Listing 5.3: VM Configuration

```
import os, re
arch = os.uname()[4]

kernel = "/usr/lib/xen-default/boot/hvmloader"
builder='hvm'
memory = 2048
maxmem = 5120
shadow_memory = 8
name = "webserver1"
vcpus = 2
maxvcpus = 2
vif = [ 'bridge=virbr0,mac=02:16:3e:10:11:10' ]
disk = [ 'file:/home/ramesh/webserver1.img,hda,w']
#,'file:/home/ramesh/ubuntu-12.04.5-desktop-amd64.
iso,hdc:cdrom,r' ]
device_model = '/usr/lib/xen-default/bin/qemu-dm'
boot="c"
#boot="d"
vnc=1
vnclisten="0.0.0.0"
vncconsole=0
vfb = ['type=vnc,vnclisten=0.0.0.0,vncpasswd=test123,
vncdisplay=2,keymap=no']
acpi = 1
apic = 1
sdl=0
stdvga=0
serial='pty'
usbdevice='tablet'
#on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

The VM's are configured in the second server *Server2* so the disk image files are all put in there. Primarily for testing purpose we installed all of the VM's in both of them. However, later we have used *Server1* for testing purpose and closed all the VM's because of saving the storage.Therefore the overall infrastructure of our research project has been stated in the figure 5.2.

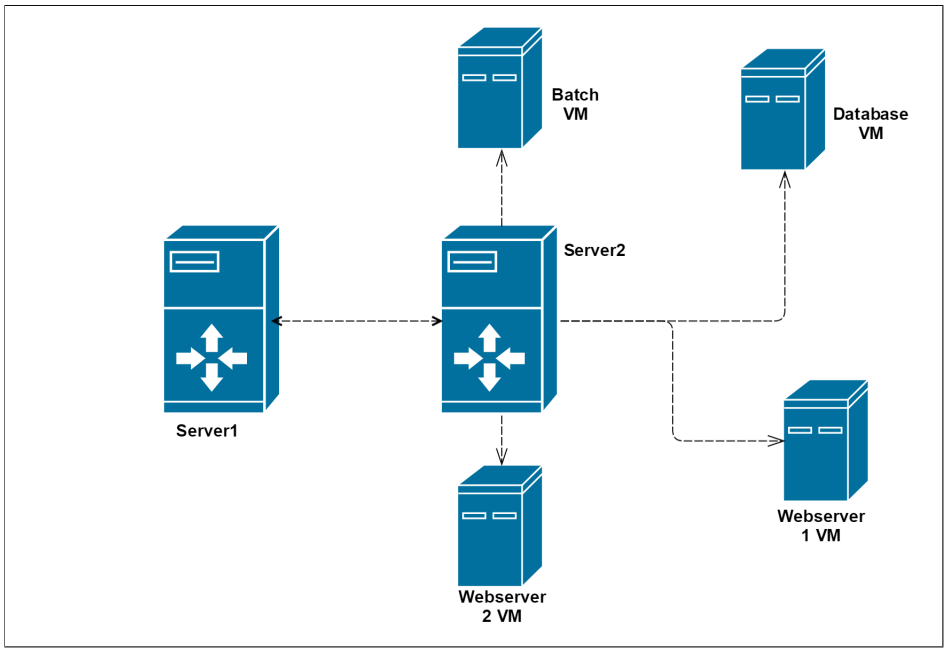

Figure 5.2: Overall Infrastructure

5.4 Overview of Experiment:

To perform the experiment we divided the whole infrastructure into three parts They are *Client side*, *Server side* and *Control side* [3].

5.4.1 Client Side:

Client side is basically the portion where the workload pattern is feed into the *HTTPMon* or *Autobench* and then simulate the traffic by sending HTTP requests. Afterwards, it estimated the response time based on the sending requests.

5.4.2 Server Side:

In server side all the VM's are running except the database VM. Because, controller adjust the resources all the VM's and allocated them based on the instruction. Only database server does not have the elastic resources so that it cannot perform in this operation. In addition the initial domain *Dom0* also running in the server side. Load balancing tool such as HAProxy, bulletin board application RuBBoS also set up as a server side application [3]. The bootup procedure of the second webserver also take place in server side to perform the horizontal scaling.

5.4.3 Control Side:

The controller works in the control side and performing the vertical or horizontal scaling. The traffic arrives in the control side and distributed

between the webserver VM's. Controller runs at a specific time interval and collect the data based on controller metrics for real time and non real time application. According to the collection of data the decision has been made that whether to increase or decrease the resources. However, the decision has been made whenever the resource utilization is more than 80% otherwise no actions has been taken by the controller itself. In addition control side is also responsible to allocate the resources of the batch VM. If all the resources utilized the resources then it is going to check the resources of the batch VM. If the batch use more resources then the control side can allocate the resource by stealing the resources from the batch. Whenever all the resources has been utilized then the controller send the signal to the server side to boot up a new webserver to perform the horizontal scaling.

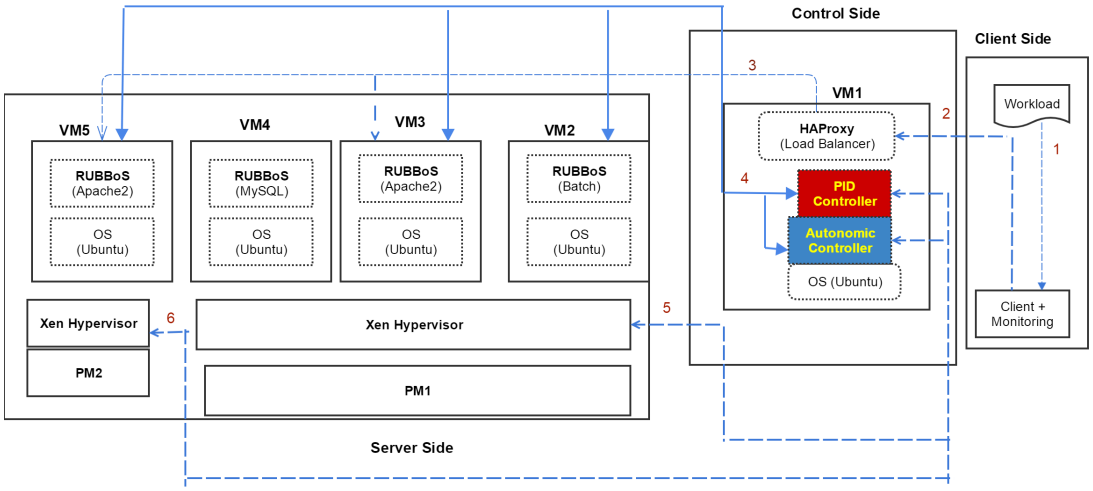

Figure 5.3: Experimental Overview

5.5 HAProxy

HAProxy is a load balancing tool, which is used to balance the load between the webservers. In the case of vertical scaling scenario there is only one webserver VM has been working. But whenever all the resources are utilized and another webserver is boot up to perform horizontal scaling then load balancing is needed to distribute the load between the two web VM's. An algorithm *roundrobin* has been designed which distributes the traffic between the two webservers. We choose HAProxy because HAProxy has native support for SSL which makes configuration lot easier than the other load balancing applications. The default port 1947 is enabled which gives a real time overview about the statistical data about webservers [3]. The configuration file of HAProxy is listed in list 5.4.

Listing 5.4: HAProxy Configuration

```
listen  appli2-insert 0.0.0.0:10002
        option  httpchk
        balance roundrobin
        cookie  SERVERID insert indirect nocache
        server  inst1 192.168.114.56:80
        cookie server01 check inter 2000 fall 3
        server  inst2 192.168.114.56:81
        cookie server02 check inter 2000 fall 3
        capture cookie vgnvisitor= len 32

        option  httpclose
        rspidel ^Set-cookie:\ IP=

listen  appli3-relais 0.0.0.0:10003
        dispatch 192.168.135.17:80

listen  appli4-backup 0.0.0.0:10004
        option  httpchk /index.html
        option  persist
        balance roundrobin
        server  inst1 192.168.114.56:80
        check inter 2000 fall 3
        server  inst2 192.168.114.56:81
        check inter 2000 fall 3 backup
```

5.6 RUBBoS:

RUBBoS is a bulletin board benchmark application and it was developed by the project JMOB [63]. The main task of RUBBoS is that it enables the end users to browse and post their comments and activities. In spite of lots of advantages, the setting up procedure of RUBBoS is bit difficult task. To perform the installtion of RUBBoS the first step is to install the webserver, *MySQL* database server and initialize the *MySQL* database. *MySQL* is required to install the *PHP* [63].

RUBBoS is a well structured benchmark application and it requires *PHP5* version to make it usable. The first step is to install a web service platform in both of the webserver VM. To employ *PHP5* a web service platform is needed. In this case *Apache2.0* is installed with the module *Apache MPM prefork* because of it is an open source web service platform, thread safe and more suitable for *PHP5* applications. Furthermore, a timeout has been set for 5 seconds, the parameter has been names as *Keep Alive timeout*. To setup the MySQL database we have to deploy a separate VM where Mysql database is installed. The *PHP5* configuration file of RUBBoS is listed in list 5.5.

Listing 5.5: PHP Script

```
<?php
function getDatabaseLink(&$link)
{
  $link = mysql_pconnect("10.0.0.9","bilal","oslo123");
  mysql_select_db("rubbos", $link) ;
}

function getMicroTime()
{
  list($usec, $sec) = explode(" ", microtime());
  return ((float)$usec + (float)$sec);
}

function printHTMLheader($title)
{
  include("header.html");
  print("<title>$title</title>");
}

functionprintError($scriptName,$startTime,$title,
$error);
{
  printHTMLheader("RUBBoS ERROR: $title");

  printHTMLfooter($scriptName, $startTime);
}

function authenticate($nickname, $password, $link)
{
  $result = mysql_query("SELECT id FROM users WHERE
  nickname=\"$nickname\" AND password=\"$password\"",
  if (mysql_num_rows($result) == 0)

  $row = mysql_fetch_array($result);
  return $row["id"];}

function getUserName($uid, $link)
{
  $user_query = mysql_query("SELECT nickname
  FROM users WHERE id=$uid", $link);
  $user_row = mysql_fetch_array($user_query);
  return $user_row["nickname"];
}
?>
```

5.7 HandBrakeCLI:

For doing the Batch job *HandBrakeCLI* tool has been configured. To perform the configuration of the *HanfBrakeCLI* tool we have to boot up a completely separate Virtual Machine(VM). *HandBrakeCLI* basically a video encoding to tool that can encode video for instance 1000 kbps for *MPEG* video and 160kbps audio. It is also CPU intensive and able to perform in multiprocessing [3]. The command of HandBrakeCLI is enlisted in list 5.6.

Listing 5.6: HandBrakeCLI

```
HandBrakeCLI -i Source -o Destination
```

5.8 Workload Pattern:

Workload has been generated to test the system can properly handles the workload and allocate the resources. To simulate the workload pattern we have to generate only a single type workload that is the *trend* based workload. The *trend* based workload has been generated gradually started from 0 clients to 1000 and then stabilizes. The controller we have designed can capable to handle the workload which increases gradually with the amount of time. In the figure we have stated a sample of *trend* workload pattern. The requests has been increases 0 to 10000 within 10 minutes.

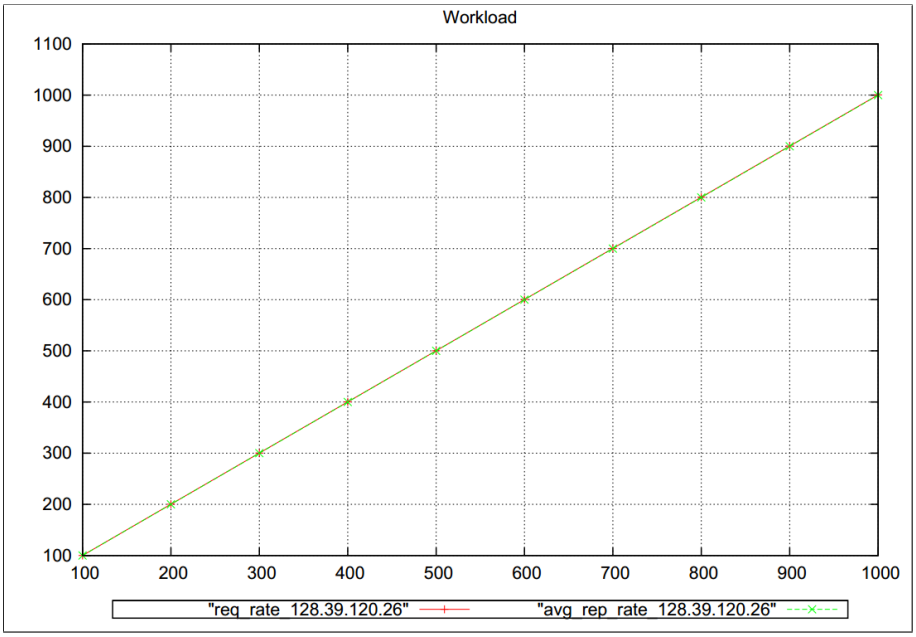

Figure 5.4: Trend Based Workload

5.9 Implementing Controlling System:

The PID controller mechanism has describe in detail in section 3.3. Also how PID controller mechanism is perform the operations along with the existing autonomic controller was highlighted briefly in section 4. Both of the controller has been implemented in a python script. The existing autonomic controller model and script both are taken from the the research conducted by *Bilal et al.* [3]. The autonomic controller script has been listed in Appendix. The PID controller class file is listed in list 5.7.

Listing 5.7: PID Class

```
def __init__(self, P=1.3, I=0.3, D=0.3, Derivator=0,
Integrator=0, Integrator_max=500, Integrator_min=-500):
    self.Kp=P ,self.Ki=I,self.Kd=D
    self.Derivator=Derivator
    self.Integrator=Integrator
    self.Integrator_max=Integrator_max
    self.Integrator_min=Integrator_min
    self.set_point=0.5
    self.error=0.0
 def setPoint(self,set_point):
        self.set_point = set_point
        self.Integrator=0
        self.Derivator=0
def setIntegrator(self, Integrator):
    self.Integrator = Integrator
def setDerivator(self, Derivator):
        self.Derivator = Derivator
def setKp(self,P):
      self.Kp=P
def setKi(self,I):
        self.Ki=I
def setKd(self,D):
        self.Kd=D

    def getPoint(self):
        return self.set_point

    def getError(self):
       return self.error

    def getIntegrator(self):
     return self.Integrator

    def getDerivator(self):
        return self.Derivator
```

PID controller script has several parameters such as *Kp,Ki and Kd* which are manually configured in the script. The controller runs in the interval of every ten seconds for webservers and every five minutes for the batch processing job [3]. Autonomic controller script code there is several built in functions as well as several parameters for instance to count size of size of CPU and memory two parameters cpu_i and mem_i. Besides these two there are more parameters have been defined in code for controlling the resources which is inside the autonomic controller script. Resource parameters is defined for each and every VM's so that the batch job can allocate 10 vcpu [3] . They are highlighted in table 5.3. Initially the system has 10vcpu and 12GB of memory and a buffer has set for used memory 512MB. Because of if memory reduces less than 512MB the whole system is going to crash.

Control Parameter	
Control Parameter	Value
minimum-vCPU	1
maximum-vCPU	10
minimum-memory(MB)	12000
minimum-used-memory(MB)	1024
url1	http://10.0.0.12/PHP/RandomStory.PHP
url2	http://10.0.0.10/PHP/RandomStory.PHP

Table 5.3: Controller Parameters

To assure better QoS an algorithm has been done to perform the vertical scaling mechanism. In vertical scaling mechanism will be perform in the fist stage of the experiment. Afterwards, if all the vCPU and Memory is allocated then bootup a new VM to perform horizontal scaling. Memory and vCPU is allocated with the script with some certain commands of Xen hypervisor. The commands are also listed in list 5.8. The main reason of performing the horizontal scaling to assure the desired level of Quality of Services(QoS), if the first Physical Machine has the lack of resources then it can perform in the second Physical Machine [3].

Listing 5.8: XM commands to adjust Resources

```
xm vcpu-set webserver1 [domain-id] 16[count in cores]
xm mem-set webserver1 [domain-id] 12000[count in MB]
```

A decision algorithm has been prescribed for the controlling system. The algorithm set the decision and make the resource allocation much more efficient. Algorithm gives a clear vision about how can we allocate the resources. The system basically needs 2 vcpu and 1 GB of memory else, it cannot be perform the operations, sometimes the whole system functionality completely going down. Therefore by taking all these cases in

our research we can manage to design a sophisticated and simple algorithm for our resource allocation and also consider to assure desired level of QoS.

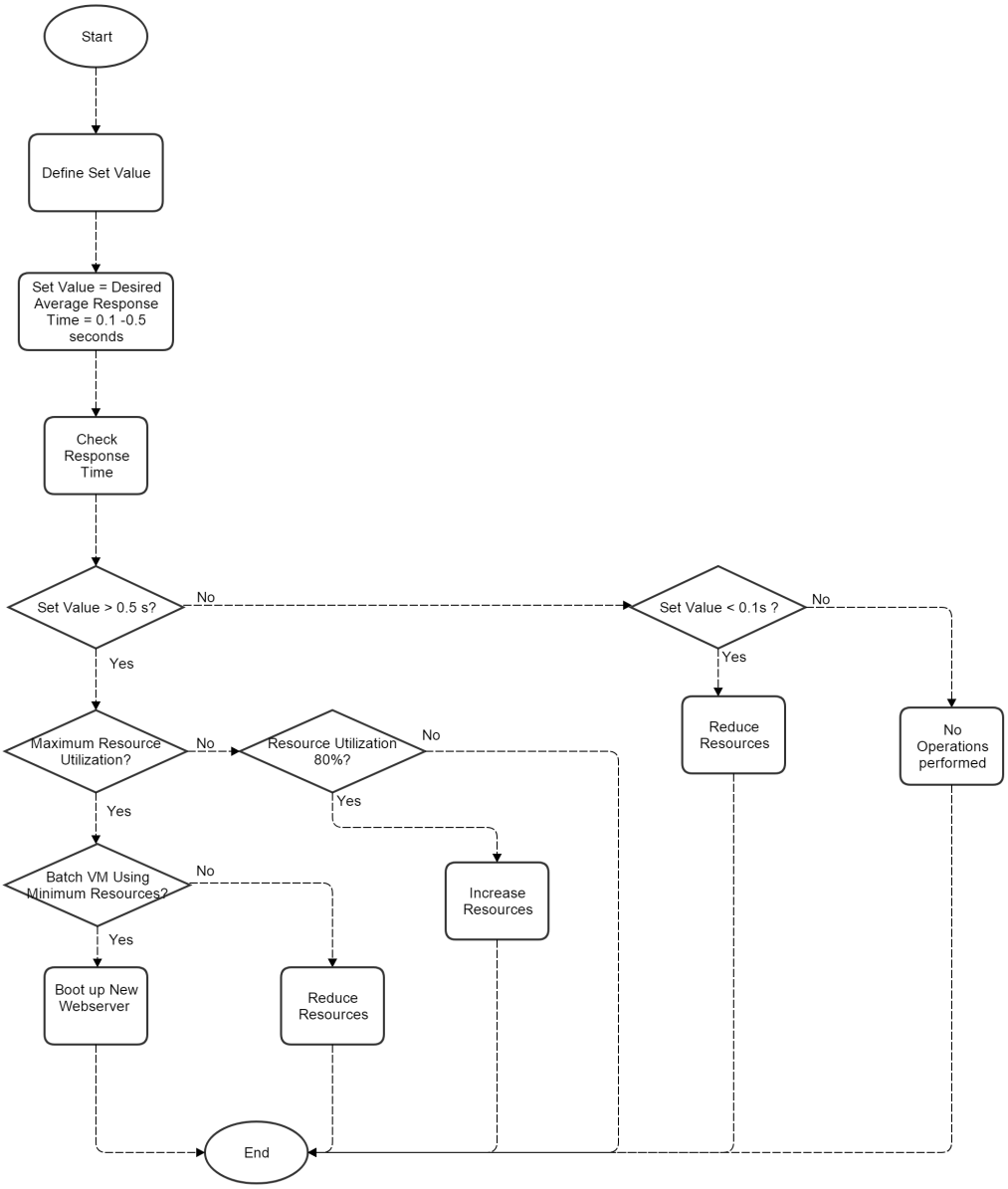

Figure 5.5: Activity diagram of decision making algorithm

The algorithm basically used by the PID controller to perform the actions. Algorithm is known as decision making algorithm because based on the instruction of the algorithm whole control system is going to perform the resource allocation. The algorithm has been designed for making decision for both real time and non real time applications. Decision making algorithm is prtially borrow from the thesis conducted by *Bilal et al.*, the algorithm also works for our system as well.

5.10 Autobench:

Autobench is using to generate the workload. The tool has been installed in local machine and generate the traffic from the local machine to the server. Therefore, to measure the external workload it is undoubtedly pne of the easiest tool. The tool also have some built in library and it also helps the graphical representation of the whole data.

Listing 5.9: Autobench

```
autobench --single_host --host1 128.39.120.26 --uri1
/index.html --quiet --low_rate 10 --high_rate 100
--rate_step 10 --num_call 10 --num_conn 1000
--timeout 5 --file random.tsv
```

Chapter 6

Measurements and Analysis

This chapter covers all the analysis phase of the experiment we have conduct. The experiments shows that how the control system is working so far. We sent traffic to the server2 from our external machine. Experiments show how the control system are going to handle the workload and allocate the resources. First phase we have counted response time in 10 seconds and 20 seconds of interval. The name of the experiment named *Control Interval*.

After performing the *Control Interval* experiments we are going to perform the experiment of *Vertical Scaling* experiment as well as *Horizontal Scaling* experiment. The experiments are going to represent graphically and prepare for the final analysis procedure. Experiments are going to reveal how our prescribed control system able to handle the situations. In addition it is going to tell us how efficient our prescribed algorithm.

6.1 Control Interval:

Control Interval experiment is the very first experiment we are going to perform. The experiment has to perform in two phases. Response time has been generated for every ten seconds of interval. Moreover, the workload has been generated nearly one thousand number of connections. In this experiment we observe the response time. The experiment is divided in two parts. They are following:

1. 10 seconds
2. 20 seconds

10 seconds of time we create workload from the local machine of end user and create workload to the physical machine and check the response time. The script has given 10 seconds of interval which means it is running each 10 seconds of time period. In addition the script will work for nearly 5 minutes. For checking the time a date function has been estimated which can calculate the time. After 5 minutes it stops automatically and generate the result. After getting the result we plot a graph with the response time and time duration. The graph also represent the resource allocation both

vCPU and Memory. First graph represent the scenario the changes of response time influence the change of vCPU.

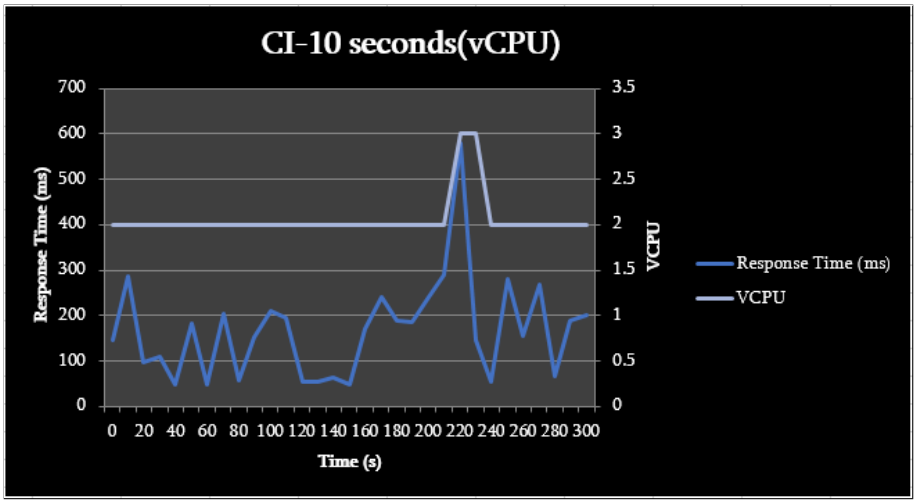

Figure 6.1: Response Time in every 10 seconds with vCPU

In figure 6.1 we have got thirty two reading in ten seconds of interval. The vertical line represents the response time in miliseconds and the number of vCPU. However, the horizontal line represents the time duration of the experiment. In the scenario it reveals that response time lies in the range of 100 to 300 millisecond. The most important scenario is that whenever the response time is increase above 500 miliseconds the vCPU increases immediately to make the response time within range. On the other hand whenever the response time is getting low and fall below 100 miliseconds then vCPU decreases. Nevertheless, to tune in the whole system at least 2 vCPU is running always, otherwise the system is vulnerable and going to crash at any time. Therefore, the system does not reduce the vCPU whenever the response time drops below 100 miliseconds.

The graph in the figure 6.2 is based on the same data of response time but in that case it represents the change of the memory in this time period. In case of memory we have to put a buffer of 512 MB so that we fixed the minimum memory usage is 1024 MB. In addition every iteration it check the memory like VCPU and allocate the memory according to the existing algorithm.

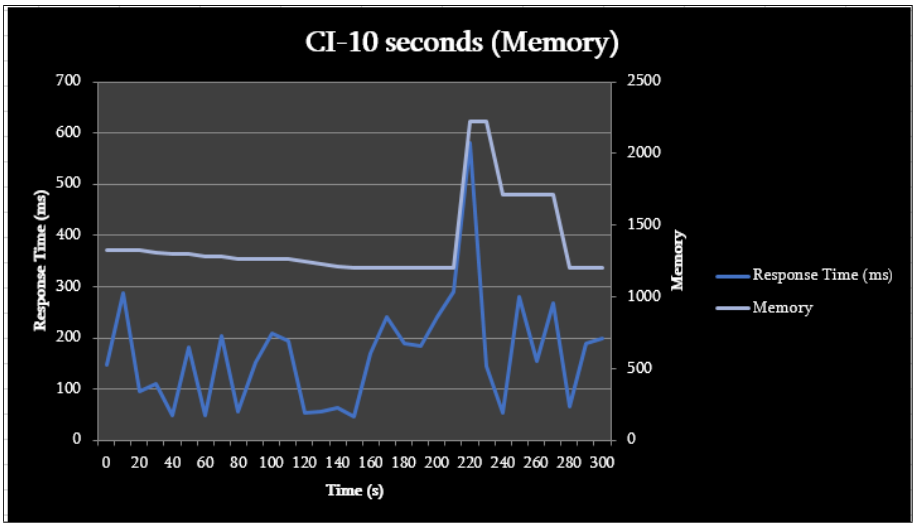

Figure 6.2: Response Time in every 10 seconds with Memory

Like vCPU the memory is also increased 1 GB whenever the response time rises above 500 miliseconds and alleviate the response time. Whenever the response time drops below 100 miliseconds the memory drop 512MB but there is a loophole in this system. Response time drops so frequently then the used memory decreases so fast and destabilize the entire system. Therefore we have set a buffer of 512 MB and make the reduction of memory in three phases.

Next phase we make the time interval of 20 seconds, that means the response time generates in every 20 seconds. The workload is also remain same as before in between 100 to 1000 number of connections. We also do it from outside the network. Also compare the average response time of this two scenario. Figure 6.3 represent the response time which is generated in every 20 seconds and allocated the vCPU accordingly.

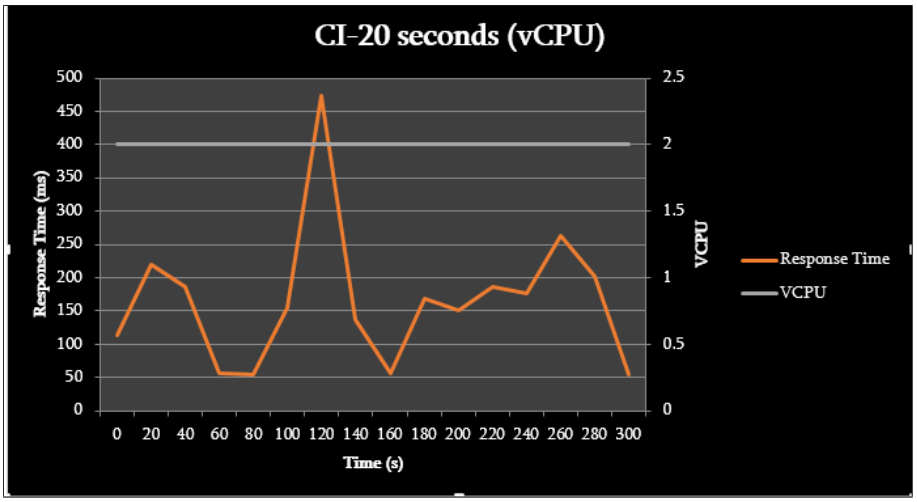

Figure 6.3: Response Time in every 20 seconds with vCPU

Figure 6.3 the response time lies in between 56 to 475 miliseconds. The highest peak is the 475 miliseconds, like figure 6.1 response time drops at the same rate like before.

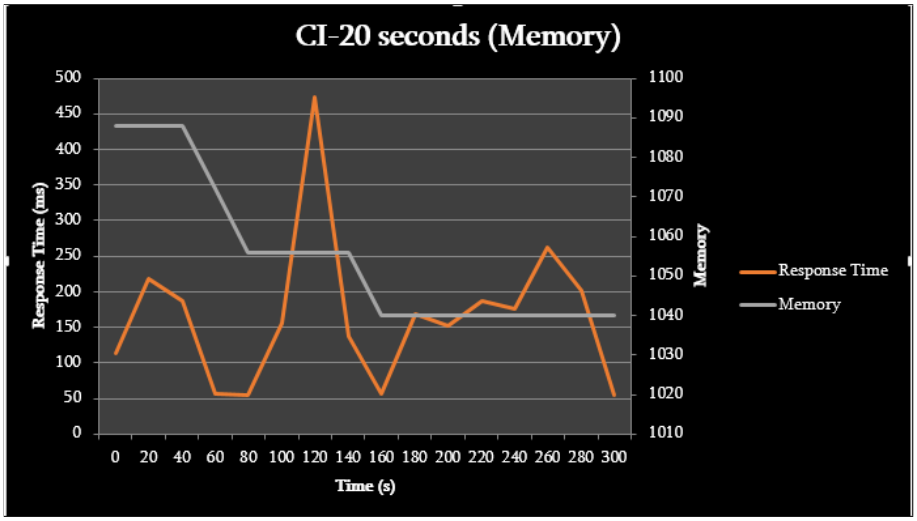

Figure 6.4: Response Time in every 20 seconds with memory

Figure 6.4 the fluctuation rate of memory is graphically represented. Response time starts from 113.77 milliseconds and initially 10088 MB memory has been allocated. In the graph it clearly visible that there is no sudden peaks which is more than 500 milliseconds. Therefore there is no increase of memory usage. However, the response time drops below than 100 milliseconds and memory has been reduced same as 10 seconds of interval and the lowest memory usage is 1040 MB which is pretty decent usage of Memory.

6.2 Webserver(Vertical Scaling) :

The vertical scaling procedure is perform with only one webserver and tackle the workload for 0 to 2000 clients. The script is running for 5 minutes and trend based workload pattern has been applied. Moreover, the 10 seconds of control interval has been used to perform the operation. The traffic has been generated through the Autobench. While performing the operation the webserver2 has been shutdown for the proper results.

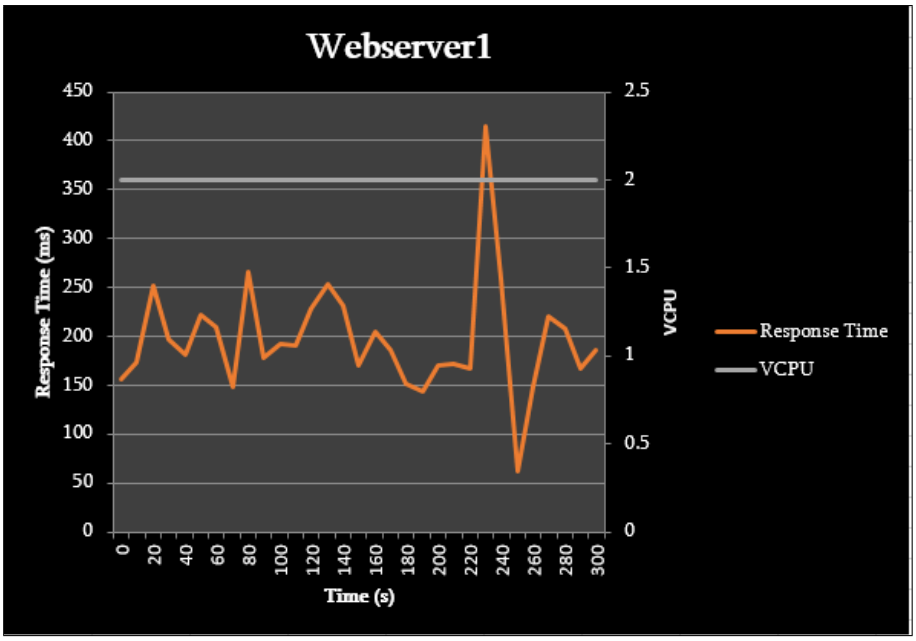

Figure 6.5: Response Time relations of webserver1 with vCPU

The maximum response time for webserver1 is 415.35 milliseconds which is less than the 500 so no more resources are allocated by the system. On the other hand only one time response time drops below than 100 milliseconds which is 62.35 milliseconds. Hence the system utilized minimal number of CPU cores so that the CPU core is not decreased by the system itself. Therefore the number of CPU core is remained static in this case. Although the vertical scaling deals allocate the nodes of the system for instance CPU and memory but we only represent the changes of the vCPU, changes of memory is also done by the system but it's graphical representation is much time consuming so forth we cannot capable to present the memory fluctuation rate in graphically.

6.3 Webserver (Horizontal Scaling):

Horizontal scaling occurs when the workload increases so fast and it's duration is much longer than original period and no other memory and CPU is remaining to handle the workload. Then a new webserver is boot up in the second physical machine in our case it is server1 and tackle the extra workload. The same workload pattern has followed in this experiment as well. The webserver2 is booted up and deploy with the existing webserver. The load has been distributed with *HAProxy*. However, when the second new webserver is spawn up and the traffic has been distributes with the round robin algorithm. Meanwhile, if the traffic increases then the webserver increase it's full resources to tackle the large amount of workload. Whenever, the situation tends to be normal the webserver1 give it's available memory to batch job and webserver2

also closes the operation and the system goes back to the vertical scaling procedure. Moreover, the second physical machine is closed all the operations of the webserver2.

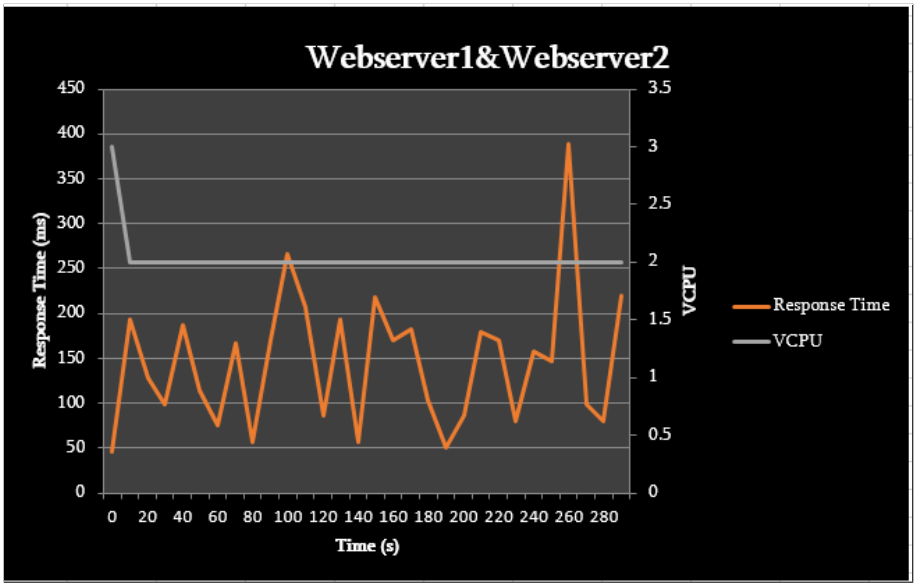

Figure 6.6: Response Time relations of webserver1 and webserver2 with vCPU

The figure 6.6 also depicts the whole situation, in the very beginning of the graph the webserver2 handles the workload with 3 CPU core and the response time drops less than 100 milliseconds then it reduces it's resources.

6.4 Batch Processing Experiment:

The experiment of batch processing jobs is bit easier, it is more CPU intensive rather than the memory, whenever the system needs memory than it first check that it uses minimal amount of resources or not. We established batch VM to perform the experiment and allocate 2 vCPU for resource utilization. The batch VM does not need to provide extra amount hence it is not memory intensive.

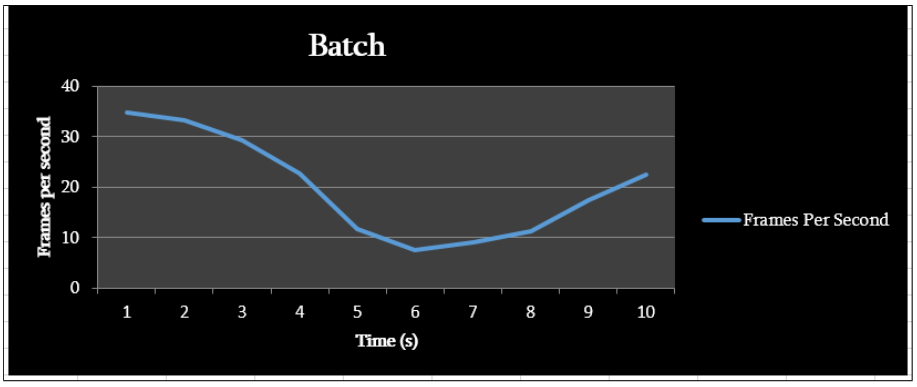

Figure 6.7: Frames per Second for Batch Process

6.5 Analysis:

In this sections the overall analysis of the experiments are presented and evaluate them accordingly. Also perform comparison about how the control system behave in all of the experiments. In addition how much vCPU and memory utilized in all the experiments we have performed so far.

6.5.1 Control Interval:

Response time fluctuation rate in 10 seconds of time interval is bit higher than the 20 seconds of interval, the highest response time in 10 seconds of interval is 579.59 milliseconds on the other hand the highest response time in 20 seconds interval is 474.36 milliseconds. However, the lowest rate of response time lies on the 10 seconds of interval which is 47.16 milliseconds. In tables 6.1 the highest and lowest rate of response time of both of the scenario is represented. Furthermore, the average response time in 10 seconds of interval is only 1.63 milliseconds higher than the 20 seconds of time interval. However, in both of the cases 0 to 1000 HTTP request has been sent to the server and trend based workload pattern has been followed.

Metrics of 10 seconds and 20 seconds interval		
Metrics	10 seconds Control Interval	20 seconds Control Interval
Average Response Time	167.80	166.17
Minimum Response Time	47.16 ms	54.17
Maximum Response Time	579.59 ms	474.36 ms
Total HTTP Request	1000	1000
Highest rate of request per seconds	100	150

Table 6.1: Difference of Metrics in 10 seconds and 20 seconds Control interval

The CPU and memory utilization also vary in 10 seconds and 20 seconds of control interval. CPU and memory utilization in 10 seconds control interval is represented graphically. Almost 70.02% CPU and 42.08% memory has been utilized in 10 seconds of control interval.

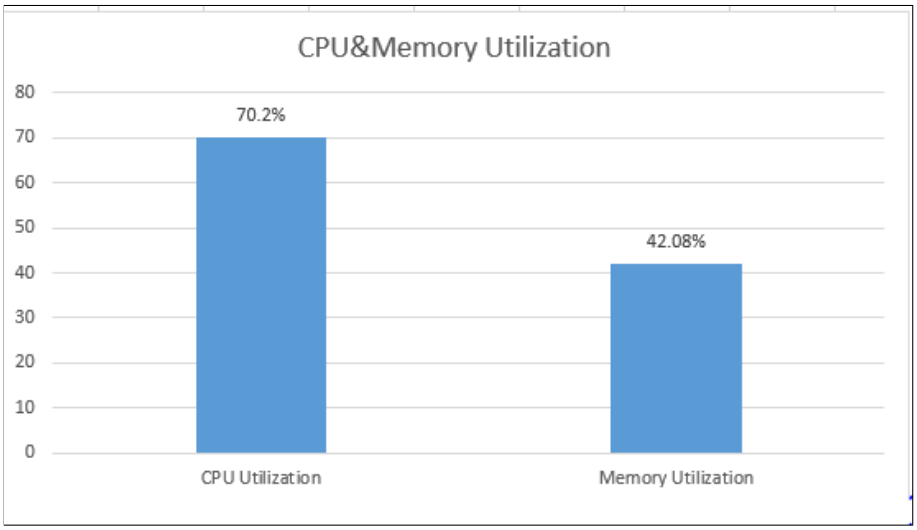

Figure 6.8: CPU and Memory Utilization for 10 seconds

In case of 20 seconds control interval the difference is not that much large than the 10 seconds of control interval. In 20 seconds of interval 67.9% of CPU has been utilized the difference is only 2.3% also in memory the difference is much lower than CPU utilization it is only 0.11%. Therefore the data reveals that the amount of CPU and memory utilization is almost same in both of the cases.

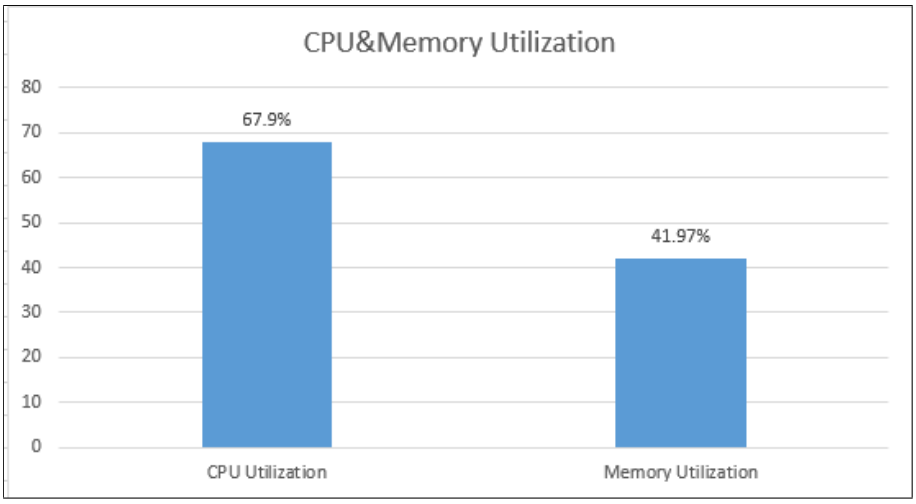

Figure 6.9: CPU and Memory Utilization for 20 seconds

6.5.2 Webserver(Vertical Scaling):

In the vertical scaling phase 0 to 2000 HTTP requests has been generated with the tool *Autobench*. The average response time is also calculated and that is 197.08 milliseconds. The minimum response time is 62.35 milliseconds and the highest response time is 415.35 milliseconds. Table 6.3 has been listed the data of the vertical scaling.

Metrics of Webserver1	
Metrics	Webserver1
Average Response Time	194.08 ms
Minimum Response Time	62.35 ms
Maximum Response Time	415.35
Total HTTP Request	2000
Highest rate of request per seconds	500

Table 6.2: Metrics of Webserver1

The webserver1 CPU and memory utilization has been stated in Figure 6.10. The CPU and memory utilization of webserver1 is 42.70% and 39.65%.

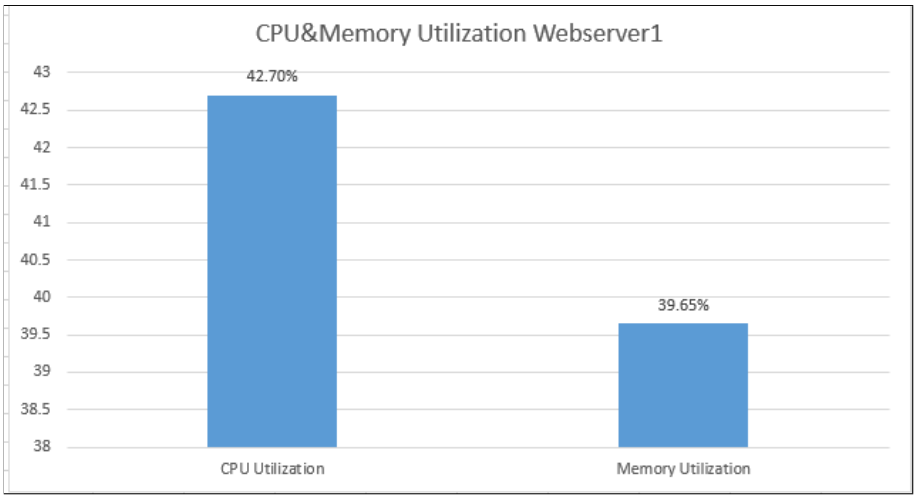

Figure 6.10: CPU and Memory Utilization for Webserver1

6.5.3 Webserver(Horizontal Scaling):

In the horizontal scaling phase the same amount of workload has been like the previous experiments, also *trend* based workload pattern has been followed. We also generate 0to 2000 HTTP requests with the tool *Autobench*. The minimum response time is 45.46 milliseconds and maximum response time is 389.21 milliseconds. Also the average response time is 48.12 milliseconds less than vertical scaling.

Metrics of Webserver2	
Metrics	Webserver1
Average Response Time	145.96 ms
Minimum Response Time	45.46 ms
Maximum Response Time	389.21 ms
Total HTTP Request	2000
Highest rate of request per seconds	500

Table 6.3: Metrics of Webserver2

The webserver2 CPU and memory utilization has been stated in Figure 6.11. The CPU and memory utilization of webserver1 is 38.70% and 24.87% which is bit lower than webserver1 because of webserver1 performing both in vrtical and horizontal scaling and webserver2 is only perform in horizontal scaling.

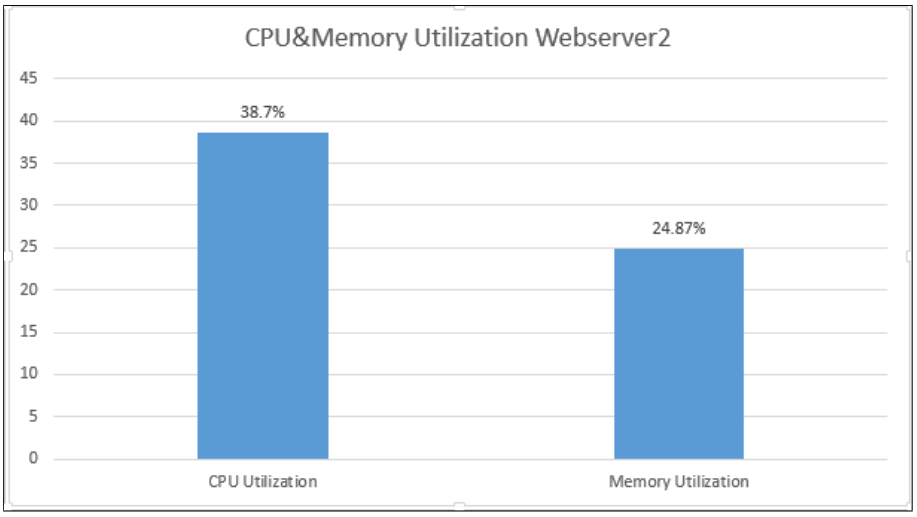

Figure 6.11: CPU and Memory Utilization for Webserver2

6.5.4 Batch Processing Experiment:

In the case batch processing the lower has higher priority which means frames > 15 fps is counted as SLA violation and if the frames > 20 fps then the process is fast and resources will be reduced. In our case the minimum frame is 7.64 fps which is counted as SLA violation and the maximum frames are 34.73 fps. Whenever it drops less than 15 the PID allocate the resource instantly to tackle the frames. However, it finally manages to make the frames per second in desired level which is in between 15 fps to 20 fps. After performing the experiment the CPU decreased from 2 to 1.

Metrics of Batch-Processing	
Metrics	Batch
Average Frames per Second	19.99
Minimum Frames per Second	7.64 fps
Maximum Frames per Second	34.73 fps

Table 6.4: Metrics of Batch-Processing

Chapter 7

Discussion:

The aim of this thesis has been to compare the overall performance of the control system using the Proportional Integral and Derivative Controller or PID controller and maintain a desired level of Quality of Services(QoS). In addition maintain the code of Service Level Agreement (SLA) also a large issue of this entire research work. In this chapter we discuss about the entire goal of the research work, what challenges we have faced so far to implement the control system and how it can improve in the near future. Furthermore it will take us to a fare conclusion about the entire project. Also increase the vCPU from 1 to 2 to tackle the increased frames.

7.1 The Problem Statement:

Firstly, the thesis work lies to implement PID controller logic with the existing autonomic controller. The research questions that we have defined in the *Introduction* chapter provide a clear indication to the reader that what we are going to achieve throughout our overall experiment. We describe the problem statement in the *Approach* chapter in detail. However, in this chapter we just going to review it once again for the convenience of the reader.

1. *How can we determine a threshold value known as set value?*
2. *How can we build a dynamic PID controller considering the differences between the set point and measured value, and apply the correlation based on proportional, integral, and derivative terms?*
3. *How can we design and build a sophisticated Service Level Agreement (SLA) driven infrastructure to provide efficient QoS by applying better resource allocation?*

In order to achieve desired Quality of Service(QoS) we defined a set value which is the average response time and average Frames per Second. In addition the set value is the measurement criteria of the controller for resource allocation. PID controller make decision on the basis of the set value and capable to build the entire control systems in a proper way

which cannot break the code of Service Level Agreement also ensure better resource allocation and make the infrastructure more efficient and robust.

7.2 Evaluation:

Results from the proposed solutions in this research work have disclosed some interesting findings. The controller parameters tuning has been determined several time and the parameters such as *Kp*, *Ki* and *Kd* are correctly determines to manage the desired level of performance. The mathematical function of PID controller which contributes in decision making is precisely generate valid Controller output and able to synchronize the existing autonomic controller mechanism. It reduces the CPU and memory utilization in a notable figure. However, the entire control system is so much robust that it can handle any amount of workload and remove the *peaks* of response time also the resources properly. Therefore, the code of service Level Agreement has been properly maintained. It also save the redundancy of the extra resources. In the analysis phase we observe the webserver CPU and memory utilization is below the 50% which ensure that it saves almost 20% of resources and ensure quality performance of the webserver.

7.3 Challenges:

The biggest challenge we have faced to implement the PID controller and coordinating the controller with the existing autonomic controller. The parameters value does not set dynamically which is one of the major challenges to implement the controller. So forth we have to set the parameters manually in every time which is really a tough task for a system administrator. To synchronize both the control system operation together we cannot do the *hot-plugging* experiment to map the VM's. Moreover the PID controller except fuzzy logic is more old fashioned and less robust to determine the external environment. The python script of the autonomic controller which allocate the resources based on the PID output is quite difficult to implement. There is no enhanced python library to implement the controller script easily. In addition the PID cannot determine the *spike* based workload, so if the workload increases randomly the controller cannot take the proper measure. Therefore, in our experiment we can deal only the *trend* based workload. Therefore, the decisions only has been made of by only a single pattern of workload. The configuration of VM's are also quite difficult task hence The Logical Volume Manager(LVM) procedure is risky and less convenient so we have to follow the disk image procedure which is very much time consuming.

7.4 Constraints:

The research work is basically focus to increase the server utilization for interactive and non interactive applications. Basically we want to minimize the response time for interactive applications with the control systems that using the PID. However, the delay time of PID impacts largely in this research. Almost ten seconds of delay has been counted to test the mechanism. Therefore we generated response time in every 10 seconds instead of the smaller value like 5 seconds or less. In addition controller has the lower stability rate.

Live Migration also puts a key limitations of this research work. Initially the plan was to perform the horizontal scaling by live migrating VM's into the second physical machine. However to implement this procedure is very handy and it required a shared-storage to be created where the disk-images had to be placed. In addition NFS share was created to accommodate the VMs on the running Physical machines. Meanwhile in the experimental phase, the VMs started to break down due to errors in memory, when the live-migration was performed.

Workload pattern has been stated in the *Challenge* section, Like in this case only the *trend* based workload has to be performed. To create more realistic workload pattern for longer period it really needs stable the web traffic since it varies during the day with a specific amount of peaks. Moreover, the controller also plays a biggest role to create the workload pattern.

The algorithm we prescribed it is based on both real and non real time applications but being time constraint we cannot manage our work for batch processing, so the algorithm has not been tested properly. However, the design of the algorithm should be more precise and sophisticated which can give better performance for decision making in both real time and non real time applications.

Fot time constraint we cannot perform the hotplugging operations like add or remove VCPU or memory which can map the VM's. It reflects how much time it takes to map the VM by adding or removing CPU or memory.

7.5 Future Works:

There is always a scope for potential development for the research work. Lots of feature and functions can be developed in order to improve the performance and expand the capabilities of the entire research. There is following areas which we have plan to develop in near future.

7.5.1 Batch Processing Files:

The non interactive applications for instance, batch processing files, need to improve for getting the better performance from the control systems. Controller need to implement more features in order to allocate resources

for bath jobs. Batch jobs are normally CPU intensive, but it needs minimal amount of memory to perform the tasks. PID controller mechanism is sometimes fail to find the suitable state for batch jobs. In near future we have to find a suitable algorithm for batch processing jobs.

7.5.2 Machine Learning:

Machine learning has to be implemented to analysis the environment. Suppose the behaviour of the system for instance which time server has to face the workload for instance during the day there is normally more traffics, so forth the controller allocate more resources from the very beginning. In addition controller has the prediction capabilities and more efficient in the peak detection. For instance control system has the capability to predict the whole scenario of environment.

7.5.3 Fuzzy Logic:

Fuzzy logic is also implemented to dynamically tuned the parameters of the PID controller as well as the autonomic controller. Implementation of Fuzzy mechanism will be a new challenge to improve the overall performance of the controllers. Fuzzy algorithm has the capability to predict the scenario of the adopted controllers.

Chapter 8

Conclusion:

The main goal of this thesis was to investigate how the vertical and horizontal elasticity improves the server utilization in large data centers. The main purpose is to allocate resources efficiently. We have to design a control system which can handle the increased amount of workload and ensure the proper usage of resources.

The problem statement was also addressed a control system which was developed with PID and Autonomic Controller which can make the decision based on the defined criteria for instance average response time and average frames per second. While the measured value of response time is more than average response time for the latency critical applications then it is allocated resources or if the measured value for is less than the average response time then it reduce the resources. A decision making algorithm also been designed on the basis of the decision making criteria.

The main contribution of this paper is that we have implemented a sophisticated control system which can capable of to allocate the resources by monitoring the resource usage. PID controller mechanism is implemented with a hybrid controller model and synchronize together to build a robust control system which can capable of decision making and allocating resources by elasticity approach for instance vertical and horizontal elasticity.

The analysis phase also give a clear indication that the performance of the controller is relatively well in resource allocation. For non real time applications for instance batch job does not perform quite efficiently but so far it can predict how the resource should allocate for the batch processing files.

Further testing and development will require to make it more reliable, secured and robust. In addition to perform elasticity in non latency based crtitcal application can add another dimension in this entire research work.

Chapter 9

Appendix

Listing 9.1: Code of Control Systems

```
 import requests
import commands
import urllib
import datetime


def cpu_logical_cores_number(domain_name):
    s = commands.getstatusoutput("sudo xm list
    | awk '/"
    + domain_name + "/ {print  $4}'")
    p = s[1]
    return p


def over_all_cpu_usage(domain_name):
    s = commands.getstatusoutput("sudo
    xentop -b -d 1 -i 2 |
    awk '/" + domain_name + "/ {print  $4}'")
    p = s[1]
    plist = p.split('\n')
    return plist[1]


def memory_usage(domain_name):
    s = commands.getstatusoutput("sudo xm list |
    awk '/" + domain_name + "/ {print  $3}'")
    p = s[1]
    return p


''' time interval in second '''
time_interval = 5
```

```
url1 = "http://10.0.0.12/PHP/RandomStory.php"
url2 = "http://10.0.0.10/PHP/RandomStory.php"
desired_response_time = 0.5

domain = 'Domain-0'
max_memory = 20484
max_cpu_core = 16

min_memory = 1024
min_cpu_core = 2

def response_time(desired_resp, r_url):
    roundtrip = 0
    try:
        start = datetime.datetime.now()
        nf = urllib.urlopen(r_url)
        page = nf.read()
        nf.close()
        end = datetime.datetime.now()
        roundtrip = (end - start).total_seconds()
        return roundtrip*10

    except requests.exceptions.Timeout as f:
        print "High Response Time"
    return roundtrip
server_params = [
    # {'name': 'Web Server 1', 'url':
    'http://google.com', 'd/omain': 'webserver1'},
    {'name': 'Web Server 1', 'url':
    'http://10.0.0.12/PHP/RandomStory.php',
    'domain': 'webserver1'},
     {'name': 'Web Server 2', 'url': '
     http://10.0.0.10/PHP/RandomStory.php',
     'domain': 'webserver2'}
]
"""Server Params"""
# for item in server_params:
#     print('stage: init')
#     Qdomain = item['domain']
#     print(Qdomain)
#     commands.getstatusoutput(
"sudo xm mem-set "
+ Qdomain + " 20000")
#     commands.getstatusoutput("
sudo xm vcpu-set "
+ Qdomain + " 16")
```

```
try:
    # while True:
    print('++++++++++++++++++++++++++++++++')
    for item in server_params:
        name = item['name']
        url = item['url']
        domain = item['domain']
        print('For ' + name + ' :')
        res_time = float(response_time
        (desired_response_time, url))
        #time = requests.get(url,
        timeout=max_response).
        elapsed.total_seconds()
        #print "This time", time
        core_number = int(cpu_logical_cores_number
        (domain))
        cpu_usage = float(over_all_cpu_usage(domain))
        memory_in_use = int(memory_usage(domain))
        print ('Interval:')
        print('Response time : ' + str(res_time))
        print 'Number Of Logical CPU cores : '
        + str(core_number)
        print 'Usage Of TOTAL CPU : ' + str(cpu_usage)
        ' %'
        print 'Memory : ' + str(memory_in_use) +'MB'
        if res_time > 0.5:
            print('Increasing Resource')
            if not (core_number + 1) >
            max_cpu_core:
                commands.getstatusoutput("
                sudo xm vcpu-set
                " + domain +
                " " + str(core_number + 1))
                print('Updated Number of Core :' +
                str(core_number + 1))
            else:
                print('Max number of
                CPU Cores are being
                used')
 if not (memory_in_use + 1024) > max_memory:
                commands.getstatusoutput("sudo
                xm mem-set "
                + domain +
                " " + str(memory_in_use + 1024))
                print('Updated Memory :' +
                str(memory_in_use
                + 1024)
                + ' MB')
```

```
            else:
          print('Max amount of memory is being used')

        if res_time < 0.1:

            if not (core_number - 1) <min_cpu_core:
                print('Reducing CPU')
                commands.getstatusoutput("sudo xm
                vcpu-set
                "
                + domain +
                " " + str(core_number - 1))
                print('Updated Number of Core :' +
                str(core_number - 1))
            else:
            prrint('Min number of CPU Cores are
            being
                used')

            if (memory_in_use - 512) > min_memory:
              print('Reducing Memory')
             ands.getstatusoutput("sudo xm mem-set
                + domain
                + " "
                + str(memory_in_use - 512))
                print('Updated Memory :' +
                str(memory_in_use
                - 512)
                + ' MB')
  else:
      nt('Min amount of memory is being used')
except KeyboardInterrupt:
    print('Exited Successfully')
```

Bibliography

[1] Tarek Abdelzaher et al. 'Introduction to control theory and its application to computing systems'. In: *Performance Modeling and Engineering*. Springer, 2008, pp. 185–215.

[2] Tarun Agarwal. *The Working Principle of a PID Controller for Beginners*. URL: https://www.elprocus.com/the-working-of-a-pid-controller/.

[3] Haugerud Ahmad Yazidi and Farokhi. 'Orchestrating Resource Allocation for Interactive vs. Batch Services using a Hybrid Controller'. In: 2016.

[4] Karl Johan Astrom. *Control System Design*. 2002.

[5] Derek P Atherton and S Majhi. 'Limitations of PID controllers'. In: *American Control Conference, 1999. Proceedings of the 1999*. Vol. 6. IEEE. 1999, pp. 3843–3847.

[6] Aytekin Bagis. 'Determination of the PID controller parameters by modified genetic algorithm for improved performance'. In: *Journal of Information Science and Engineering* 23.5 (2007), pp. 1469–1480.

[7] *Bandwidth Extension*. URL: https://en.wikipedia.org/wiki/Bandwidth_extension.

[8] Paul Barham et al. 'Xen and the art of virtualization'. In: *ACM SIGOPS operating systems review*. Vol. 37. 5. ACM. 2003, pp. 164–177.

[9] Md Faizul Bari et al. 'Data center network virtualization: A survey'. In: *IEEE Communications Surveys & Tutorials* 15.2 (2013), pp. 909–928.

[10] Thomas C Bressoud and Fred B Schneider. 'Hypervisor-based fault tolerance'. In: *ACM Transactions on Computer Systems (TOCS)* 14.1 (1996), pp. 80–107.

[11] Sean Carlin and Kevin Curran. 'Cloud computing security'. In: (2011).

[12] Dan Chen and Dale E Seborg. 'PI/PID controller design based on direct synthesis and disturbance rejection'. In: *Industrial & engineering chemistry research* 41.19 (2002), pp. 4807–4822.

[13] Guillon Christophe. *Program Instrumentation with QEMU*.

[14] *Citing a web page with no author*. URL: https://wiki.xenproject.org/wiki/Xen_Project_Schedulers (visited on 13/10/2016).

[15] *Citing a web page with no author*. URL: https://wiki.xen.org/wiki/Credit_Scheduler (visited on 13/10/2016).

[16] *Citing a web page with no author*. URL: https://www.xenproject.org/directory/directory/projects/92-rt-xen.html (visited on 2013).

[17] Louis Columbus. 'Roundup Of Cloud Computing Forecasts And Market Estimates, 2016'. In: *Forbes Magazine* (2016).

[18] *Control Systems*. URL: http://www.iitg.ernet.in/engfac/chitra/notes/EE350/Lecture-14.pdf.

[19] J Ding et al. *Pqemu: A parallel system emulator based on qemu. ICPADS*. 2011.

[20] *Disturbance Rejection*. URL: http://www1bpt.bridgeport.edu/~risc/html/proj/introb/node16.html.

[21] Kit Eaton. 'How One Second Could Cost Amazon $1.6 Billion In Sales'. In: *Fast Company* 14 (2012).

[22] Fahimeh Farahnakian et al. 'Using ant colony system to consolidate vms for green cloud computing'. In: *IEEE Transactions on Services Computing* 8.2 (2015), pp. 187–198.

[23] Mohammad Al-Fares, Alexander Loukissas and Amin Vahdat. 'A scalable, commodity data center network architecture'. In: *ACM SIGCOMM Computer Communication Review*. Vol. 38. 4. ACM. 2008, pp. 63–74.

[24] Soodeh Farokhi et al. 'A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach'. In: *Future Generation Computer Systems* 65 (2016), pp. 57–72.

[25] Soodeh Farokhi et al. 'Coordinating cpu and memory elasticity controllers to meet service response time constraints'. In: *Cloud and Autonomic Computing (ICCAC), 2015 International Conference on*. IEEE. 2015, pp. 69–80.

[26] Michael Fenn et al. 'An evaluation of KVM for use in cloud computing'. In: *Proc. 2nd International Conference on the Virtual Computing Initiative, RTP, NC, USA*. 2008.

[27] SÃ«bastien Christophe FrÃ«dÃ«rique FrÃ«mal, Michel Bagein and Pierre Manneback. 'Optimizing Xen Inter-domain Communications'. In: *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*. ACM. 2015, pp. 3–10.

[28] Marius Gligor and FrÃ«dÃ«ric PÃ«trot. 'Combined use of dynamic binary translation and systemc for fast and accurate mpsoc simulation'. In: *1st International QEMU UsersÃ¢ÂÂ Forum*. Vol. 1. 2011, pp. 19–22.

[29] Chunye Gong et al. 'tg'. In: *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*. IEEE. 2010, pp. 275–279.

[30] Eugene Gorelik. 'Cloud computing models'. PhD thesis. Massachusetts Institute of Technology, 2013.

[31] Albert Greenberg et al. 'The cost of a cloud: research problems in data center networks'. In: *ACM SIGCOMM computer communication review* 39.1 (2008), pp. 68–73.

[32] Albert Greenberg et al. 'VL2: a scalable and flexible data center network'. In: *ACM SIGCOMM computer communication review*. Vol. 39. 4. ACM. 2009, pp. 51–62.

[33] Nikolay Grozev and Rajkumar Buyya. 'Multi-cloud provisioning and load distribution for three-tier applications'. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 9.3 (2014), p. 13.

[34] Fang-fang Han et al. 'Virtual resource monitoring in cloud computing'. In: *Journal of Shanghai University (English Edition)* 15.5 (2011), p. 381.

[35] *HandBrakeCLI*. URL: https://wiki.archlinux.org/index.php/HandBrakeCLI.

[36] Samah Sabir M Hassan and Shadi MS Hilles. 'Enhancing Security Concerns in Cloud Computing Virtual Machines:(Case Study on Central Bank of Sudan)'. In: (2014).

[37] *Hot-Plugging CPU's to conserve power*. URL: http://linuxforthenew.blogspot.no/2013/01/hot-plugging-cpus-to-conserve-power.html.

[38] Joo-Young Hwang et al. 'Xen on ARM: System virtualization using Xen hypervisor for ARM-based secure mobile phones'. In: *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*. IEEE. 2008, pp. 257–261.

[39] *Introduction to PID control*. URL: http://beta.machinedesign.com/sensors/introduction-pid-control?utm_test=redirect&utm_referrer=https%5C%3A%5C%2F%5C%2Fwww.google.no%5C%2F.

[40] Yashpalsinh Jadeja and Kirit Modi. 'Cloud computing-concepts, architecture and challenges'. In: *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*. IEEE. 2012, pp. 877–880.

[41] Pooyan Jamshidi, Aakash Ahmad and Claus Pahl. 'Autonomic resource provisioning for cloud-based software'. In: *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM. 2014, pp. 95–104.

[42] Reece Johnston et al. 'Xen Network Flow Analysis for Intrusion Detection'. In: *Proceedings of the 11th Annual Cyber and Information Security Research Conference*. ACM. 2016, p. 18.

[43] Lori M Kaufman. 'Data security in the world of cloud computing'. In: *IEEE Security & Privacy* 7.4 (2009).

[44] Jacek Kobus and Rafal Szklarski. 'Completely Fair Scheduler and its tuning'. In: *draft on Internet* (2009).

[45] Ewnetu Bayuh Lakew et al. 'Towards faster response time models for vertical elasticity'. In: *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE Computer Society. 2014, pp. 560–565.

[46] Yanfei Li et al. 'An online power metering model for cloud environment'. In: *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*. IEEE. 2012, pp. 175–180.

[47] *Logical Volume Manager*. URL: https://en.wikipedia.org/wiki/Logical_Volume_Manager_(Linux).

[48] M Malathi. 'Cloud computing concepts'. In: *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*. Vol. 6. IEEE. 2011, pp. 236–239.

[49] *Margaret Rouse*. URL: http://searchcio.techtarget.com/definition/horizontal-scalability.

[50] Peter Mell, Tim Grance et al. 'The NIST definition of cloud computing'. In: (2011).

[51] Ankur Mishra et al. 'Cloud computing security'. In: *International Journal on Recent and Innovation Trends in Computing and Communication* 1.1 (2013), pp. 36–39.

[52] *Mrinmoy Ghosal*. URL: https://www.linkedin.com/pulse/20141021201313-156372715-vertical-scaling-vs-horizontal-scaling-big-data (visited on 2014).

[53] *No author*. URL: https://en.wikipedia.org/wiki/Completely_Fair_Scheduler (visited on 2013).

[54] *No author*. URL: https://en.wikipedia.org/wiki/Control_theory (visited on 2017).

[55] NRDC. *AmericaÃ¢ÂÂs Data Centers Are Wasting Huge Amounts of Energy*. Aug. 2014. URL: https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf.

[56] Diego Ongaro, Alan L Cox and Scott Rixner. 'Scheduling I/O in virtual machine monitors'. In: *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. ACM. 2008, pp. 1–10.

[57] *Oriental.net*. URL: http://www.httpmon.com/.

[58] *PID controller*. URL: https://en.wikipedia.org/wiki/PID_controller.

[59] *PID for Dummies*. URL: https://www.csimn.com/CSI_pages/PIDforDummies.html (visited on 2016).

[60] *PID Theory Explained*. URL: http://www.ni.com/white-paper/3782/en/.

[61] Radu Prodan and Simon Ostermann. 'A survey and taxonomy of infrastructure as a service and web hosting cloud providers'. In: *Grid Computing, 2009 10th IEEE/ACM International Conference on*. IEEE. 2009, pp. 17–25.

[62] *Response time*. URL: https://en.wikipedia.org/wiki/Response_time_(technology).

[63] *RUBBoS*. URL: https://github.com/michaelmior/RUBBoS.

[64] Reiner Sailer et al. 'Building a MAC-based security architecture for the Xen open-source hypervisor'. In: *Computer security applications conference, 21st Annual*. IEEE. 2005, 10–pp.

[65] RaÃēl Alves Santos et al. 'Edusca (educational scada): Features and applications'. In: *Advances in Control Education*. Vol. 7. 1. 2006, pp. 614–619.

[66] JORDAN SCHAENZLE. *PID Ã¢ÂÂ Helping Computers Behave More Like Humans*. URL: https://spin.atomicobject.com/2016/06/28/intro-pid-control/.

[67] Mina Sedaghat, Francisco Hernandez-Rodriguez and Erik Elmroth. 'A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling'. In: *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*. ACM. 2013, p. 6.

[68] *Sensitivity*. URL: https://en.wikipedia.org/wiki/Sensitivity_(control_systems).

[69] *Set Point*. URL: https://en.wikipedia.org/wiki/Setpoint_(control_system).

[70] *Steady State*. URL: https://en.wikipedia.org/wiki/Steady_state.

[71] *Steady State Error*. URL: https://www.facstaff.bucknell.edu/mastascu/eControlHTML/Design/Perf1SSE.htm#What.

[72] Subashini Subashini and Veeraruna Kavitha. 'A survey on security issues in service delivery models of cloud computing'. In: *Journal of network and computer applications* 34.1 (2011), pp. 1–11.

[73] Salesforce UK. *Why Move To The Cloud? 10 Benefits Of Cloud Computing*. Oct. 2016. URL: https://www.salesforce.com/uk/blog/2015/11/why-move-to-the-cloud-10-benefits-of-cloud-computing.html.

[74] *Understanding PID Control*. URL: http://www.controleng.com/single-article/understanding-pid-control/ebf9ab7fe3e5571f83901e0b8f3d8f07.html.

[75] Antonio Visioli. 'Fuzzy logic based set-point weight tuning of PID controllers'. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 29.6 (1999), pp. 587–592.

[76] Ingolf WaÃmann, Daniel Versick and Djamshid Tavangarian. 'Energy consumption estimation of virtual machines'. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM. 2013, pp. 1151–1156.

[77] Andreas Weber et al. 'Towards a resource elasticity benchmark for cloud environments'. In: *Proceedings of the 2nd International Workshop on Hot Topics in Cloud service Scalability*. ACM. 2014, p. 5.

[78] Kim Weins. *Cloud Computing Trends: 2016 State of the Cloud Survey*. Oct. 2016. URL: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey.

[79] *What is Disturbance Rejection?* URL: http://www.cds.caltech.edu/~macmardg/wiki/index.php?title=What_is_disturbance_rejection%5C%3F.

[80] Wikipedia. *Virtualization*. Oct. 2016. URL: https://en.wikipedia.org/wiki/Virtualization.

[81] CS Wong et al. 'Fairness and interactive performance of o (1) and cfs linux kernel schedulers'. In: *Information Technology, 2008. ITSim 2008. International Symposium on*. Vol. 4. IEEE. 2008, pp. 1–8.

[82] *Xen*. URL: https://wiki.debian.org/Xen#Domain_0.

[83] Sisu Xi et al. 'Rt-xen: Towards real-time hypervisor scheduling in xen'. In: *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*. IEEE. 2011, pp. 39–48.

[84] L YamunaDevi et al. 'Security in virtual machine live migration for KVM'. In: *Process Automation, Control and Computing (PACC), 2011 International Conference on*. IEEE. 2011, pp. 1–6.

[85] Jun Zhang et al. 'Performance analysis towards a kvm-based embedded real-time virtualization architecture'. In: *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*. IEEE. 2010, pp. 421–426.