UiO : Department of Informatics
University of Oslo

A Study of The impact of Virtualization on Computer Networks

Pratik Timalsena
Network and System Administration
University Of Oslo
Master's Thesis Spring 2013

A Study of The impact of Virtualization on Computer Networks

Pratik Timalsena
Network and System Administration
University Of Oslo

2013

Abstract

Virtualization is an imminent sector of the Information and Technology in the peresent world. It is advancing and being popuraly implemented world wide. Computer network is not isolated from the global impact of the virtualization. The virtualization is being deployed on the computer networks in a great extent. In general, virtualization is an inevitable tool for computer networks. This report presents a surfacial idea about the impact of the virtualization on the computer network. The report enlightens about the measurement of the some popular network performance metrics. The report describes the experiments which has been separately conducted on a bare metal environment and virtualized platforms. The analysis and comparison between these two experimental issue is covered on this document. The web server is a most prominent entity of the computer so that to address a little bit more scenarios, the performance measurement of the web server is explained. Thus, the report is reflection of the two fold task, measurement of the performance of the computer network along with measurement of the performance of web server. In addition, the document provides basic information about the influence of virtualization on a small private computer network.

Contents

List of Figures

List of Tables

Preface

This thesis is submitted in partial fulfilment of the requirements for a Master's Degree in Network and System Administration at University of Oslo. It is completed in four months of time period. My supervisor on this project has been Lecturer Tore Moller Jonassen. This thesis has been made solely by the author; a lot of the contents, however, is based on the research of others, the refercences to these sources have been provided as far as possible. Several persons have contributed academically, practically and with support to this master thesis. Firstly, I would like to thank my supervisor Tore Moller Jonassen for the most valuable supervision during whole master thesis. I am very much thankful to Associate professor Harek Haugerud for his support, inspiration and worthy guidelines. I would like to thank System Administrator Aamir Maqbool for his help on experimental setup.
Finally, I would like to thank my friends for being helpful and supportive during my master thesis.

Part I

Introduction

Chapter 1

Introduction

The world of Information and Technology is growing rapidly day by day. Within the advancement in computer and technology sector, Institute and Business Interprise are facing more devices, services , changes along with many more complexities and challenges. The expectation and needs of the user and consumers is reaching higher and higher [24] . Simultaneously IT researchers and Experts are working harder and finding a better and dynamic solution to address the demand of consumers.
The enormous growth of server within the data centers as per growth of enterprises within unproportional sequence lead to think about better solution to utilize and manage resources within a minimum overhead. Thus, to address such kind of critical situation virtualization comes up as a handy solution as a server virtualization or many more types of virtualization technology with specific features. [25]So virtualization is that convenient methodology of dividing the resources of the computer hardware into multiple execution environments [26]. In addition virtualization due to efficient resource management underutilized hardware is used. Consolidating servers favors the administravite cost is kept low. Moreover, it also provides flexibility to the system by allowing multiple OS and application to run in single hardware. Apart from these isssue virtualization has many more advantages as well [19]. Although Virtualization has many advantages and is regarded as very prominent technology of the current era, it does have many demerits as well. In the case of performance, the performance of virtualized system is always lower than the system or network without virtualized. Eventhough the virtual machines are isolated from the physical hardware , it is still connected to the physical hardware in the sense that failure to the hardware leads to the failure of the VM as well [1]. when we inspect closely at virtualization, we can always see both merits and demerits. So, it sounds interesting to know the overall impact of the virtualization by measuring the performance of the computer Network without virtualization and and then comparing again the performance implementing the virtualization. Thereby it gives an opportunity to study the impact of virtualization in a broad sense.
Apart from the measurement of the network perfomance on a computer network , the performance of the webserver is also measured within a

physical environment and virtual environment. The web service is a hot topic in the computer world every organization and institute has their own separate webserver on a network. So, it sounds worthy to measure performance of the web server and analyze the effect of virtualization on the performance of web server.

1.1 Problem Statement

The main objectives of the thesis is to measure the impact of the virtualization on computer network. This task is quite vague but can be limited to specific tasks. The project will be a two folded task. The separate experiments will be carried out in two different enviroment bare metal and virtual. More precisely, the main problem statements is listed as following.
i) Measurement of network performance on bare metal environment
ii)Measurement of network perfomance on virtual environment
iii) Comparision of the results from bare metal and virtual environment
iv)Figure out the impact of virtualization on computer networks.

Chapter 2

Background Review

2.1 Virtualization

Virtualization is the technology which provides an software layer between the hardware and the operating system. On the top of which many applications run. Thus, multiple guest operating system can run on the same machine thereby hiding the physical resources of the computing system from the operating system [1] . In this way virtualization contributes for improvement of resource utilization, sharing or aggregation of physical resources [1] . Virtualization has been defined in several ways due to advancement in technology or its deployment that is why it is hard to find a general definition. According to Sing in his article, Virtualization is "Framework or methodology of dividing the resource of computer hardware into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partioning, times haring , partial or complete machine solution, emulation, quantify of service and many others"[2]. Before getting in depth about virtualization a brief history about virtualization is described in next section.

2.2 Brief History of Virtualization

Virtualization was first developed by IBM corporation in 1960s to partition large mainframe computer into logical instances where single physical mainframe runs as a host. [3] This virtualization facilitated the time-sharing and resource sharing in an expensive hardware. After creation of time sharing system, during 1970s, there was further research going on about virtualization. Scrodowa and Bates demonstrated how to create virtual machine on IBM OS/360s in 1973. They also described the use of IBM's virtual machine monitor , a hypervisor, allocate memory, storage and I/O in efficient way for an efficient virtual machine Implementation. However, overhead, performance degredation and loose storage security was demerit of this model.
The time sharing system was performing well however there was a big problem, whenever there is an error in an application whole system is crashed. To overcome this unreliability problem, the application was isol-

ated from each other. To facilitate the isolation of application one system was used for one application. However, this solution turned out very expensive and resulting heavy wastage of system resource as well. For remedy of this problem one solution was implemented, the isolated application creates the copies of hardware and software for each of them so that user can run their own application which lead to the concept of virtualization [4].
In 1980s the inexpensive microprocessor based computer were available which enabled individual user to have their own machine at the affordable price. Due to this reason, the neccessity of virtualized declined. [5] centralized computing and virtual interest was substituted by individual servers performing individual function such as web, email, database etc [6] .
The micoprocessor capabilities improved during 1990s and the multiprocessor shared memory computer were introduced in the market. Due to this fact, operating system design turned out complicated and limited the speed of multiprocessor computer system. As user wanted to run multiple os in same hardware necessity of virtualization rose again. Fortunately. the researcher at the Starford University addressed the solution by reintroducing the idea of the virtualization by writing sofware for large microprocessor computer system. In 1998, they established a company called Vmware to develop and market the product dedicated to virtualization of computer which are based on Intel X86 architecture. After that many companies are also doing their job on virtualization. Thereby comes the Server virtualization which is most famous nowdays. In this way the virtualization is growing its momentum[5] .

2.3 Vital Component Related to Virtualization

Before immerging inside the classification of Virtualization a brief description of inevitable terms in virtualization is described in this section.

2.3.1 Virtual Machine

Many years ago, the idea of virtual machine was endorsed by IBM on purpose of Time-sharing and resource sharing on expensive Mainframe hardware [7]. According to Goldberg virtual machine is"A system which is a hardware-software duplicate of real existing machine, in which a non trival subset of the virtual machine's instructions execute directly on host machine. "[8] This definition clearify that the virtual machine is a simply the isolated replica of the underlying physical machine. The figure 2.1 depicts a typical virtual machine system. There is a layer of software called VMM or Hypervisor just above the layer of physical hardware. The hypervisor is discussed in more detail in another section. The hypervisor completely manages the machine hardware and creates virtual machine. we can see that each virtual machine has its own OS and applications running on it. Thus, every virtual machine is solely isolated from each otherand act like independent physical machine. In this way, the user

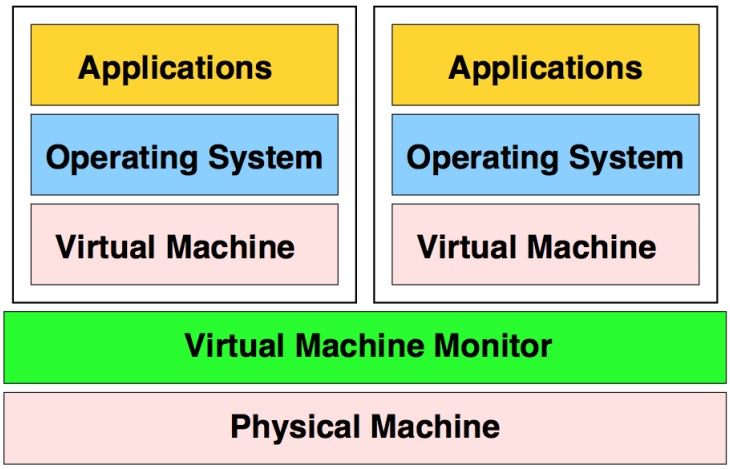


Figure 2.1: Virutal Machine System [9].

of each virtual machine are in illusion that they are having a dedicated physical machine[9] .

2.3.2 Virtual Machine Monitor

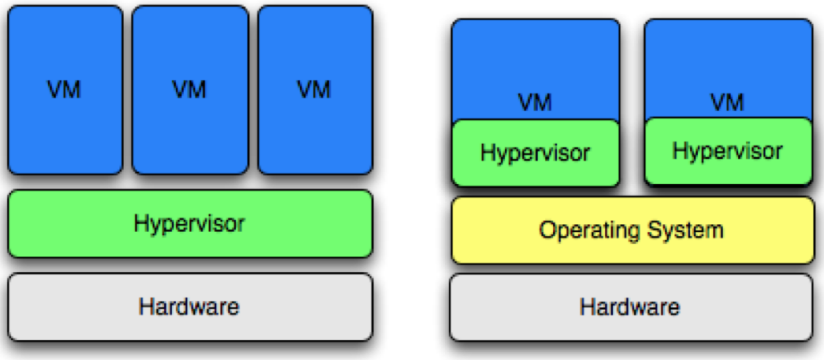
A Virtual Machine Monitor is a thin layer of software that can run directly on the hardware of a machine. It exports a replicas of underlying phyical hardware known as Virtual Machine[10]. On the other hand, VMM allows these multiple virtual machine to run on a single physical machine where each virtual runs its own operating system independently[12] . Actually VMM stands betweenone or more operating system and hardware thus making each running OS in illusion that it is controlling the hardware. Nevertheless, the VMM is controlling indeed the physical hardware and multiplexing running OSs across the hardware of the physical machine. Actually, it acts as a operating system for operating system but in a very low level provided that OS still thinking it is interacting with physical hardware. In this way, VMM facilitates the transparency[11].

2.3.3 Types of Hypervisor

There are basically two types of VMM or Hypervisor[12].

Type I Hypervisor

This type of hypervisor runs directly on the hardware without host operating system. As we see on the figure2.2(a) , all OSs run inside the virtual machine independently and hypervisor plays role to maintain interaction between physical harware and virtual machines. Xen is good example of Type I hypervisor.



(a) Type 1 (b) Type 2

Figure 2.2: Types of Hypervisor [12].

Type II Hypervisor

This Hypervisor runs on the top of the operating system of Host. In the figure2.2(b), each vm runs on the layer above the hypervisor where different operating runs on them called as guest operating system. Each guest OS runs as process in the host. Vmware and KVM are example of Type II hypervisor.

2.4 Classification of Virtualization

There are several techniques of virtualization and several software implementation including open-source and commercial. That's why it is not easy to sort out virtualization in a genric way. Nevertheless , in a broad aspect, virtualization is sorted out in 3 categories as following [14].
i) Client Virtualization
ii) Server Virtualization
ii) Storage Virtualization

Some of the virtualization's types are described in a different section in detail as following.

2.4.1 Server Virtualization

Server virtualization simply means virtualization of many servers in a single physical server. Thus, multiple virtual servers are created where and operating system is on those servers and are able to run different application on them in a same manner like a physical server. Virtual servers can be created either by dividing hardware of software. If the physical server is divided by hardware then there is a great advantage that heavy load occuring in a particular section has no effect on other sections. On the other hand, if the physical server is divided by software then the hardware

resources can be assigned and shared among multiple machine even the resource is excluseliyely used by specif virtual machine. [15]
The server virtualization can be classified in 3 catergories as followings.

operating system virtualization

Operating system virtualization is applied on servers. Operating system virtualization enables machine to host multiple operating system on it. Basically. there are two technologies being implemented in Os-level virtualization. Firstly. a single kernel is shared by containers reperesented by Virtual Machines. The other technique allows various Operating System with distinct kernels to run within a single physical Machine inside the VMs. The main advantage of this technique is that user are provided a high facility of reconfigurability[16.] As all these VMs run on a single machine sharing the same hardware resource, there should be some supervisor to manage resource sharing and provide reconfigurability to users. The supervisor in Os virtualization is called Hypervisor or Virtual Machine Manager.

Full Virtualization

Full virtualization allows multiple OS or applications to run in virtual machine without any modifications. The hypervisor need to translate and emulate every detail of physical hardware platform for achieving full virtualization [16].

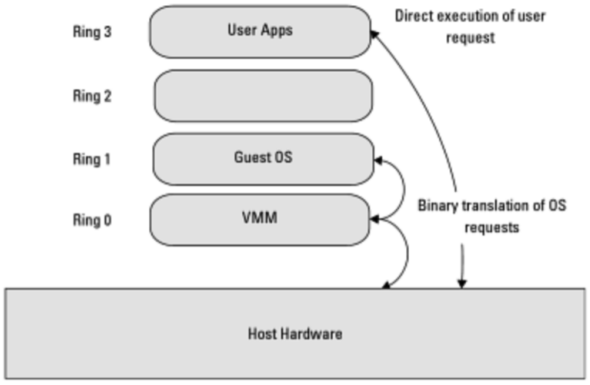


Figure 2.3: Full Virtualization[17]

As we can see in the figure 2.3, the full virtualization can be done by combination of binary translation and direct execution. The CPU of Physical machine executes nonsensitive instruction at native speed. The OS instructions are translated in a fly and stored in a cache for future reference whereas user level instruction are run without modification at native speed. Thus, Full Virtualization provides better security and isolation for VMs and also migration of VM is easier because the same instance of guest OS can run in virtualized or native hardware [17].

paravirtualization

In parvirtualization the guest OS should be modified prior to operation. It provides an interface between Host hardware and the modified guest OS. Unlike, the full virtualization the guest machine can realize that they are running in a virtual environment[1]. The striking feature of paravirtualization is Address Space Virtualization which provides each virtual machine with its own unique address space. The main advantage of paravirtualization is that it has better performance than full virtualization and also easier to implement when there is no hardware assistance available. On the other hand, guest operating system can not be run without modification in VM thereby virtual machine suffers from deficiency of backward compability and regarded as not so portable as well.

Hardware Emulation

Emulation is virtualization technique where a complete physical hardware technique is created in software [18]. The software which present an emulated hardware environment with in which guest operating system operates , is known as Hypervisor or VMM. In this technology there is a layer of hypervisor between virtual machine and hardware of physical machine. The operating system is not installed on the host rather the OS is insatalled on Virtual Machines. The VMM plays role to communicate between OS of virtual machine and hardware of Host machine. The VMM manages everything for the interaction between VMs and Host system [19]. The drawback of hardware emulation is that it has a lot of performance overhead because VMM has to translate every instruction of VMs to interact with host system. On the other hand, the guest OS and the VM are both stored in the files which forms a complete system image . Due to fact that, Vm can be easily migrated from one hypervisor to another hypervisor residing on different physical machine.

2.4.2 Storage virtualization

Storage virtualization is somehow typical resource virtualization. Firstly. physical resource which are scattered over the network is aggregated and thus forming a storage pool which leads to creation of a logical storage. Although the logical storage is created by aggregating scattered physical resource, it looks like a single monolithic storage fro end users [1].
The major advantage of storage virtualization are: It isolates application from underlying physical devices, improves availability and manintainability of system and expand storage very fast, enables to reduce down time for backup and other maintenance function thereby facilitating migration of data from system and application andsupports large storage device beyond of physical possibility [20].

2.5 Advantages and Disadvantages of Virtualization

Anything in this world posses advantages and disadvantages. They are like two sides of a coin which are inevitable. It might be benefical in one context whereas on the other context leaves disadvantages as well. In the same manner, virtualization has impressive benefits with some remarkable demerits. In this section, some of the striking merits and demerits of virtualization in the current world of computer and technology , is briefly described. Being optimistic, lets discuss the advantages of virtualization first. The followings are very common advantages of virtualization in specific context.

Server Consolidation

The individual or distrubuted servers can be virtualized and consolidated into a single server which is one of the key advantage of the virtualization. This is not just a case of moving original servers into a centeral one. The original servers are decomissioned and still facilitating users to access server application image from the new and consolidated server. Ultimately, provinding great advantage of reduced labor cost, facilitating remarkable network security saving energy and power etc.

Proper Utilization of assets

In the case of the distributed or isolated servers, they might be idle when there is no need to perform dedicated task. But when they are disposed via virtualization, there resource can be used. Thus, virtualization can be better solution for proper assessts utilization.

Reduction in Power , Heat and Cooling requirements

While using many small servers we do not care how much power and energy they are consuming. It seems ignorable but infact we are using a lot of energy for power , heat and cooling equipments. So, when these small servers being consolidated into a large server we can save energy requirement for power heat and cooling [28].

Optimization of Network load

Virtualizing the independent servers and combining to a single box can free bandwidth of the network and reduces the network load. // **Expedite Fault Islolation Measure**

When a VM is compromised by malware or gets corrupted. It is always easy to shut it down and restart newer one with uncorrupted copy of VM. This task is very easy and comfortable to perform in VM than in a physical server [28] [29].

Moving Logical Servers Between Hardware Virtualization enables to move complete virtualized server to another hardware without affecting it. So, when additional resource is needed logical servers can be moved to new hardware. This is a quite handy feature of virtualization.

Simplified and convenient Disaster Recovery Process

Virtualization presents a very handy facility to take snapshots of the state of the virtual machines while running and save as a file. That's why whenever there is problem or crash in system. The system can be restored to previous working state with the help of saved snapshot file.

Besides impressive advantages followings are major disadvantages of the

virtualization.

Increased risk with physical damage

It's nice that many physical servers are consolidated into a large one in virtualization. However we can not imagine how much information we will loose if there is fault in large consolidate physical server. If hardware failure occurs in large consolidated server, there will be severe damage . we will loose much more information than in isloated servers .

Deterioration in performance level

Virtualization is a imitation of something it is not a original. Thus , it lacks many things of the original and always compromised in some way. Although it tries to reach performance level of original one but there is always some overheads in virtualization which leads to significant deterioration in performance level.

Increased variety of Complexity

we put critical server on virtualization. It sounds we are making our task easier. Some software whch are being used is easy to manage but some are really hard to manage and are tedious to handle. So, virtualization may arouse variety of complication in server.

Real time or Near Real time requirement

In some virtualization system , the system clock temporarily lags as much as 5-10 seconds when the virtual machines are under heavy load. Generally, it is not a big issue but in case of real time application it might be a serious problem when the system does operate in real time , the real time application will not work. That is why virtualization is not feasible for the real time applications.

Bottle Neck or Queuing delays

In virtualization there are many virtual machines running on a single physical hardware. That means different OS or multiple instances of the same OS are running over there. Which results in high usage of CPU, high memory requirements and high I/O traffic . Due to this fact, it may eventually lead to bottleneck. Other reason of bottleneck is also because of many application of the VM are hitting for same resource. [27]

Support

The facility of isolating applications via virtualization may simplyfy the support. However many software are not supported on virtualization or vice-versa.

Specialized skill requirement

Now days, the management of virtualization is getting simpler but it is always demanding skilled persons who have good knowledge of virtualization at the same time. It will be disaster if the virtual environment is managed by person who do not fully understand the virtualization. That's why virtualization demands more skilled set requirements [29].

2.6 virtualization Technology

There are sever Virtualization Technology available nowdays according to need of the Institutes and companies. In this section two famous Virtualization Technology is explained.

2.6.1 KVM(Kernel Based virtual Machine)

KVM is an opensource virtualization technology for Linux on X86 hardware containing virtualization extensions(Intel VT or AMD-V). That means KVM is compatible with ntel VT or AMD-V[21]KVM is full integrated on LInux Kernel. The first Version of KVM was implemented in the 2.6.20 Linux kernel which was released in februeary 2007. Basically, KVM developers are funded by Qumarent but recently it is owned by RedHat.

KVM uses a loadable kernel module KVM. ko which facilitates core virtualization like CPU and memory virtualization. For example, KVM-intel. ko and KVM-amd. ko for Intel VT and AMD processor respectively. [15]KVM itself is not able create virtual machine that's why it needs QEMU which acts as a user-space process. Infact, QEMU is hardware emulator which is provides as OSS for emulating standard X86 PCs and other architecture. Moreover, QEMU was introduced before KVM and can operate without KVM as well.

QEMU/KVM Execution Flow

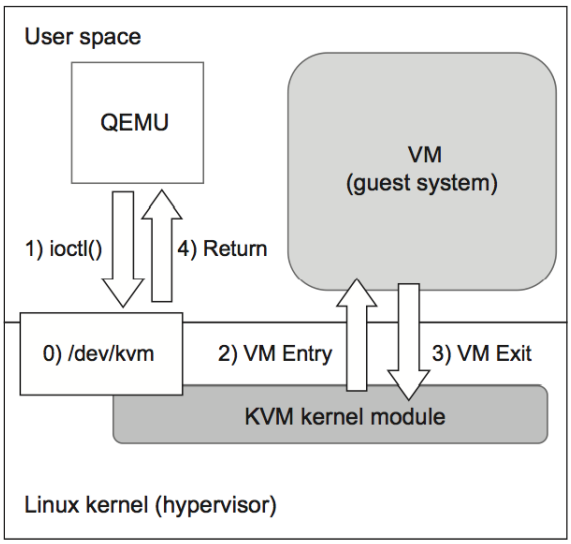


Figure 2.4: QEMU/KVM Execution Flow [22].

Figure 2. 4 explains QEMU/KVM Execution Flow. There are mainly four steps in QEMU/KVM Execution Flow. Firstly, in step 0, a file called /dev/KVM is created on KVM kernel module. This file enables QEMU to convey various requests for facilitating hypervisor function. Then QEMU

makes repeated ioct() call to instruct the KVM kernel when guest-system begins to execute on the basis of /dev/KVM file which is shown in fig as a step no 1. Ater that, kernel modules perform a VM ENTRY and executes VM ENTRY(step 2). Lastly, when the guest-system begins to execute sensitive instructions, a VM EXIT is performed and KVM kernel figures out the reason for exit. Incase, the QEMU intervention is required to execute I/O task or any other task, controll is transferred to QEMU process again(step 4) and QEMU does execution. Now, in this way controll execution reaches to the step 1 and again the process is repeated from step 1 to step 3 [22].

KVM Architecture

FIgure 2. 5 depicts the simple architecture of KVM. Here , we can see LInux kernel module as main enity of KVM architecture. The virtual machines runs like a normal Linux process as every virtual machines are linux process in KVM. These VMs are scheduled by the LInux Scheduler. Thus, KVM grabs all the benefits of Linux kernel. Hardware emulation is facilitated by modified QEMU which provides emulated BIOS, I/O bus, Network interface and disk controller [23].

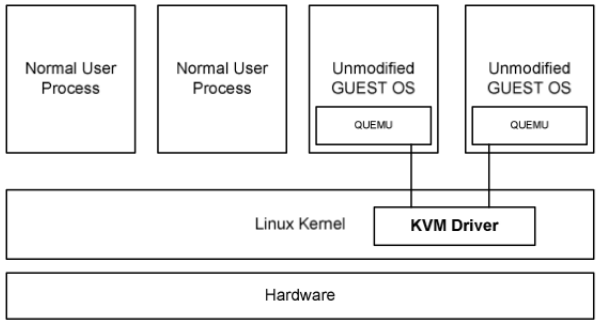


Figure 2.5: KVM Architecture [23].

KVM Related Componets

The overall stucture of KVM comprises mainly 5 components as shown in figure 2.6.

Virt-manager Thid is a GUI/CUI user interface which manages virtual machine. It also calls VM function via libvirt.

libvirt It is a tool-and-interface libarary for server virtualization software which supports Vmware ESX/GSX, Xen and QEMU/KVM.

QEMU It is basically emulator that interacts with KVM kernel module and also execute various type of guest-system processing.

KVM Kernel Module It is a sort of Linux Kernel Module capable of handling VM EXIT from guest-system and executing VM ENTRY Instructions.

Linux Kernel In KVM QEMU runs like simple process that's why sceduling of guest system is managed by Linux Kernel itself.

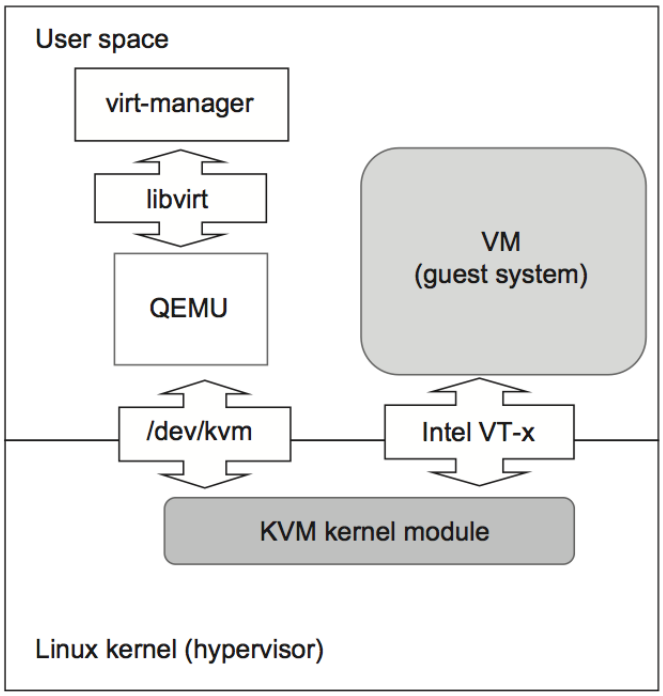


Figure 2.6: Components of KVM [21].

2.6.2 Vmware Virtualization

Vmware Virtualization is one of the popular virtualization tehnology for Intel X86 architecture. It is complete virtualization solution. Vmware Inc is the virtualization software vendor whose headquater is situated in Palo Ato, California[34]. Vmware is a unique solution of virtualization where VMM is entirely run on host operating system. Vmware installs a separate VM Driver oh the behalf of virtual machine instantained by it to provide faster and convenient access to device on the system. Basically, Vmware product comes up in market in three categories. They are Vmware Workstation, Vmware GSX Server and Vmware ECX Server respectively. The brief description of the Vmware product is given in this section.

Vmware Workstation Before describing the Vmware Workstation . It is necessary to know briefly about some important terms related to it[32].

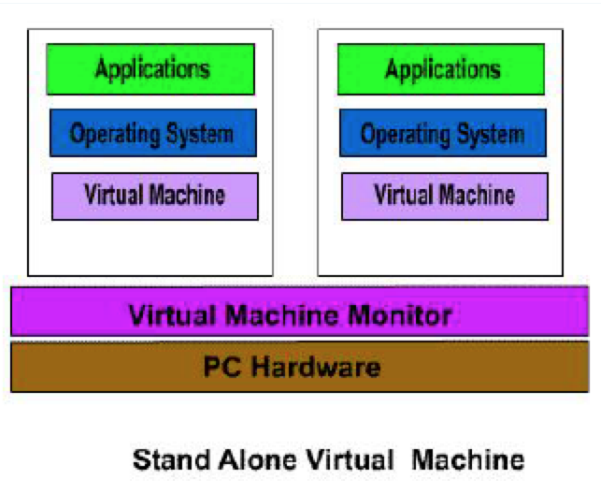


Figure 2.7: Stand alone VMM [33].

Vmware VMM It can be Stand alone or Hosted VMM. In Stand alone VMM, a software layer of VMM lies just above the bare hardware which faciliate the user to create their virtual machine. Within each Virtual machine there exists its own operating system and also application , running on the top of OS. Thus, Stand alone VMM acts almost like an Operating System of the bare metal hardware and hence does it task like memory management, resource management, I/O control etc. Figure 2.7 gives the general overview of the Stand alone Vmware VMM. Where as in Hosted VMM there is host operating system and VMM runs like an application of host operating system. Thus, Host operating system helps to manage resource, memory and scheduling etc. Hosted VMM is well depicted in the Figure 2.8

Another important term is Protection Mechanism on the Intel architecture which presents 4 priviledges level in a ring architecture as shown in Figure 2.9 below. Ring 0 Serves for Operating System , Kernel Service whereas ring 1, 2 serves for device driver and ring 3 is meant for application. Priviledgeed instructions are executed only on ring 0 otherwise protection is violated when run in anywhere else. The Vmware comprise three main components VMX driver and VMM installed in ring 0 whereas Vmware application in the ring 3. To provide high priviledge for VMM , VMX driver is installed within Operating System. VMM is loaded in Kernel with the help of VMApp. Actually, OS hsd idea about VMX driver and VMApp but it does not know about VMM. Now, there exists two world of machine one is Host world and other is VMM world. The VMM world can communicate to Host world via VMX driver . The guest OS application are executed directly on CPU via VMM. I/O operation which are priviledged are trapped by VMM and are executed on the Host world by a world switch[33]. Figure 2.10 describes detail architecture of Vmware Worksta-

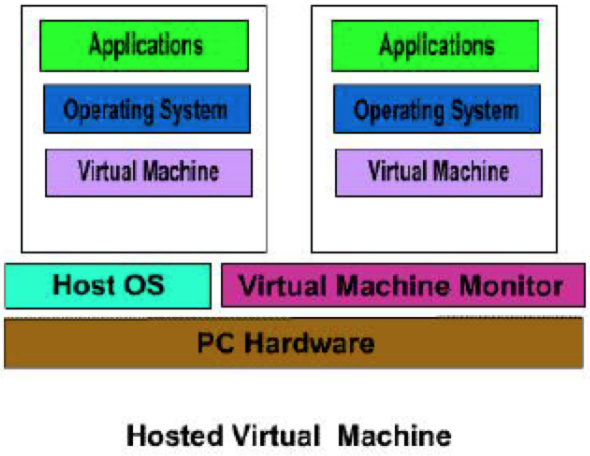


Figure 2.8: Hosted VMM [33].

tion architecture.

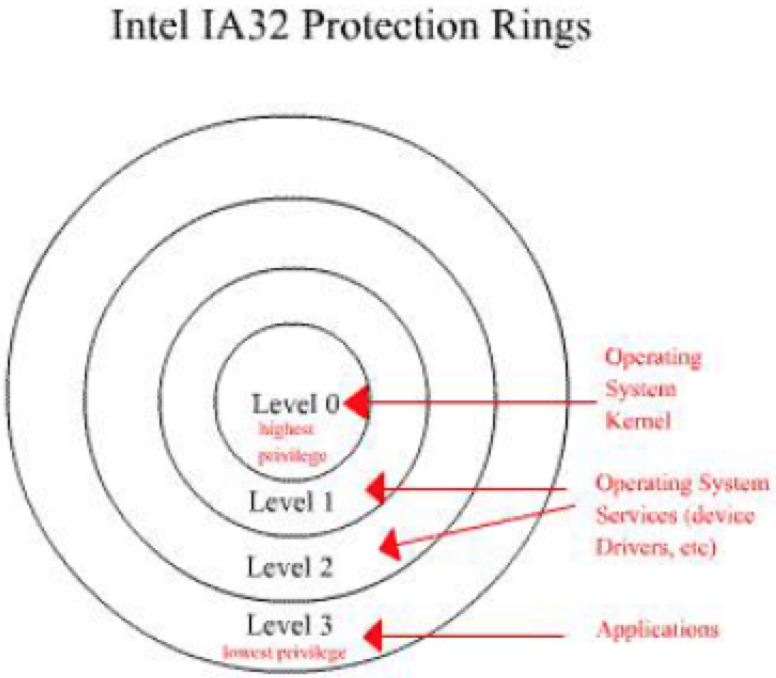


Figure 2.9: Intel IA 32 Protection Ring [33].

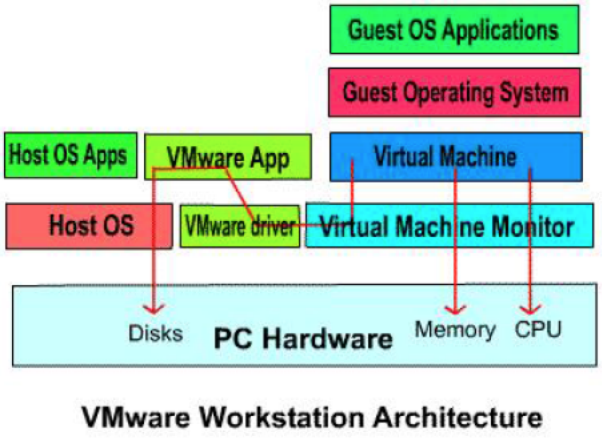


Figure 2.10: Vmware Workstation Architecture [33].

Vmware GSX In this type of Vmware solution the Host OS is already been installed on bare hardware. So, that the VMM runs as an application on a Host OS. In the figure 2.11, we can see Vmware GSX is installed on the top of the Host OS. VM are created above the Vmware GSX can communicate with host and access resource with the help of VMM. Host OS takes major responsibility for handling I/O operatins. Although Hypervisor has a compact configuration, it presents overhead and extensibility problems. Overall, it can be regarded as a optimal for PC virtualization[32].

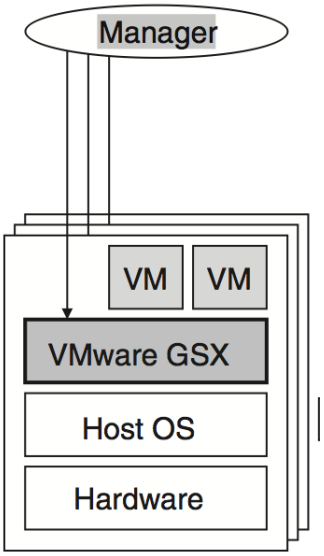


Figure 2.11: Vmware GSX [32].

Vmware ESX Unlike the Vmware GSX , Vmware ESX runs directly on the bare metal hardware without Host operating system being installed

there. Figure 2.12 depicts the general structure of VMM ESX. In the figure we can see a Console OS(COS) which facilitates Hypervisor management. Due to the fact that, it has much better performance than Vmware GSX and Vmware Workstation. The console is linux based which needs to be patched frequently which arises the security challenges. The clients are managed by Vsphere. Moreover. Vcenter in VMM ESX makes management of multiple ESX Hypervisor in much simpler way.

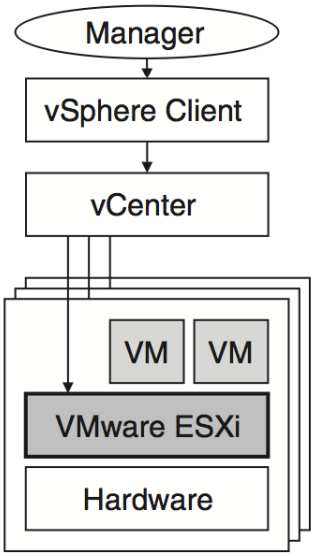


Figure 2.12: Vmware ESX [32].

Vmware ESXi The Vmware's latest Hypervisor is ESXi which do not have COS as shown in figure 2.13. Everything is almost similar to Vmware ESX but it is more compact and easy to install. Due to inavailability of COS it has not the security Challenges as well.

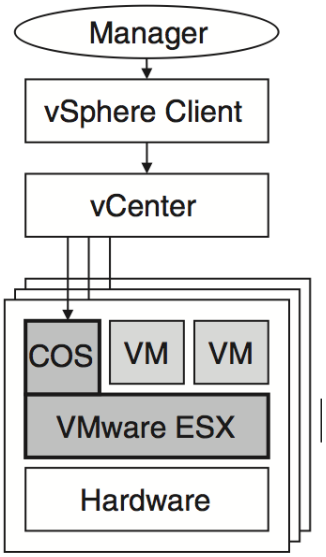


Figure 2.13: Vmware ESXi [32].

2.7 Important Metrics on Network

There are many parameters which assess the performance of the Network. It is not easy to generalize these metrics. The most common parameters which is being popularly used to measure Network performance parameters are shorltly described in this section [30] [31].

Availability

Availability measures feasibility of the network. That means percentage of time the network is running without any defect. Actually availability signifies how robust is the computer network.

Delivery

Delivery means the percentage of each service delivered without packet loss.

Loss and Error

This means fraction of packet lost while transmission or number of datagram lost due to various reason like buffer overflow. It also assess the fraction of error bits on the packet.

Delay

Delay parameter measures the time taken for a packet to make oneway or round trip from the sender to the receiver and vice-versa. Delay assess mainly three things oneway delay, round trip delay and delay variance.

Utilization

Utilization is the throughput for the link often referred as access rate.

Bandwidth

Bandwidth metrics measures the amount of the data a user can transfer through network in a time unit.

Part II

Project

Chapter 3

Planning the project

The experiments will be carried out in two different platform. Firstly, the experiment will be conducted in the Bare Metal Platform and data will be collected and presented by graphs. This will be the completion of the First stage experiment. Then the same experimental will be conducted in Virtual platfrom exactly in same manner as in the Bare Metal Platform. Very first step will be planning of hardware requirement and experimental setup. The experimental setup for the project is described in the different section more precisely. The main purpose of the project to measure the performance of the Computer network and study the impact of the virtualization on computer network. To measure the performance of the network it is very essential to find which factor of the network to measure. There are several metrics that measure the performance so it is very important to sort out and limit the metrics which will measure the performance of the computer network. Basically, the metrics like throughput, bandwidth, latency is going to be measured in the project. The more emphasis will be given to measure these parameters however if it is possible some other metrics like CPU usage, memory consumption and I/O performance of the network will be tried. More over the web server will be also measured using appropirate bechmarking tools.
On the virtualization platform context, the physical experimental set up will be imitated in the virtualization environment as far as possible. The experiment will be carried out in at least two virtualization to increase reliability of the expeimental results. The same scenario will be created and the same experiment will be carried out there. So, data will be collected and then presented by more appropirate graphs to make it more convenient to understand. Then the result of the experiments on the both platform will be analysed well. The data will be interprated and pattern will be observed carefully. Finally the experimental results of both bare metal platform and virtualized platform will be compared and after a appropirate discussion with data finally a conclusion of the experimented will be drawn.

3.1 Experimental Setup

As the experiment is going to be carried out separately in Bare metal hardware that means Sever without being virtualized and all the clients as well and within virtual environment with Virtual server and Client. There will be different experimental set up for bare metal and virtualized platform. Basically, for bare metal setup the major concern will be to build a small physical private computer network. For virtualization platfrom setup a network of virtual machine will be build up in a strong physical Server. The experimental set up for the project is described more precisely in the separate sections below.

3.1.1 Bare Metal

The experiment will be carried out on various steps . The first approach will be to build a simple private network. To collect precise data , the computer network will be setup in various steps , increasing the number of the client one by one in the very next step. To elaborate precisely, firstly the network will be set up with a server and a single client. Then the experiment will be conducted and then data will be collected. The same experiments will be conducted repeated just increasing the number of clients. In this way the network will be completed with adding all clients. There will be five clients and one server in a private Network. The network layout diagram of the computer network is depicted on the figure 3.1 and figure 3.2 below. These all client are indetical which means that they have the same hardware and software configuration . Which is one of the most important requirement for the experimental set up. In the figure we can see the simple computer

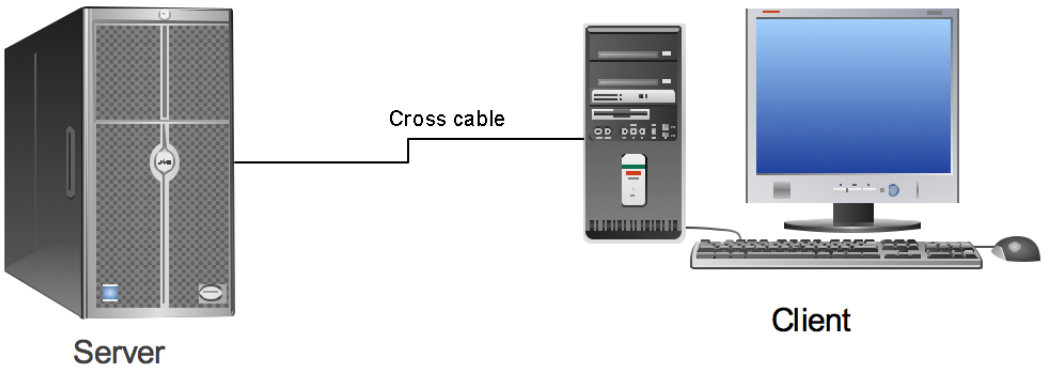


Figure 3.1: A computer Network with a single client.

network with a server and a client connected in network via a cross cable. When the experiment will be finished with this network, the network will

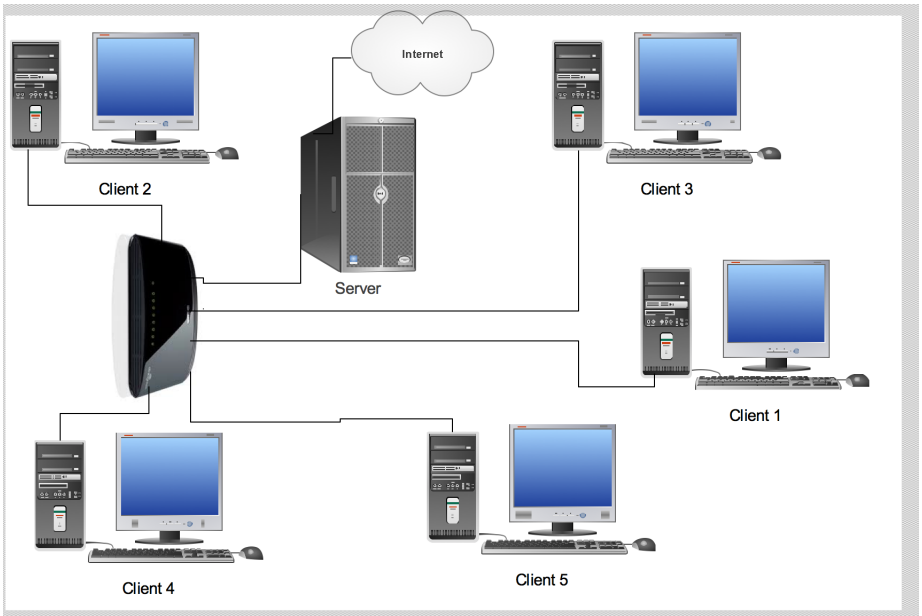


Figure 3.2: A computer Network with multiple clients.

be expanded gradually adding the clients in the network one by one. Every time the client is added the same experiment will be conducted . Finally , when there will be five clients on the network the netwok will be the final network. The computer network with a server and five client is shown in Figure 3.2 where server is connected to Internet and a private network is build with help of a switch assigning static private IP addresses to each client .

3.1.2 Virtualized Environment

For the experimental setup in virtualize platform , the powerful server which can support virtualization will be required. The virtual machines will be created which will act as a client and server and will be almost the replica of the physical network in the virtualize environment. The virtual machine will be brought in a notwork by suitable virtual network bridging. So, the virtual network bridging will be the important task to set up experiment in the virtualized platform.

3.2 Approach

At the very first step the Ubuntu server 12.10 was installed in all the machine including clients and a server. So, that the Ubuntu server 12. 10 was chosen as the appropirate operating system for the experiment. After proper designation of experimental setup the experiment was conducted according to the plan as far as possible . As per planning the experiment was carried out in two rounds. In the first round the experiment was conducted on Bare Metal setup and then on another round the experiment

was done on virtualized setup. Basically the experiment was dedicated to measure network performance along with measurement of webserver performance. The experiment is described in detail in the separate sections as follows.

3.2.1 Bare Metal

Experimental Setup

The experimental setup was designed as per planning. The small computer network was created and necessary software and application was installed. On the bare metal section everything was performed according to plan and designed which was assumed before starting the experiment.

Implementation

After compeletion of the experimental set up the experimented was carried out according to the designation . Firstly, experiment was carried out in bare metal with a computer network which has only one client and a server being connected by the a cross cable. When they were in the network and communicate with each other, they were ready to measured for network performance. To measure the network performance a handy tool called **iperf** was chosen. Actually, **iperf** measures the througput and the bandwidth of the network. After that the performance of the web server was measured. To measure the web server performance a convenient tool called **httperf** was implemented . With the help of **httperf** the average request sent by the client and the average reply sent back by the server was measured along with the average error rate. Then the same experiment was carried out just adding one more client every time. Then same experiment was carried out until the fifth client was successfully added on a network. Then the useful data was collected from the output file and saved to specific file so that it can be utilized and presented by graphs afterwards. The detail explanation of how these tools were implemented exactly in the experiment is given below in a respective section.

Implementation of iperf

The implementation of **iperf** begins with installation on both the server and client. The interesting fact about **iperf** is that it can be used to measure the both TCP and UDP bandwidth and throughput perfomance. That is why the test was conducted on both TCP and UDP modes. To use server simply **iperf** was run in both server and client. Actually when **iperf** was run on the server mode it was waiting simply for connection and requests from clients. On the other hand when **iperf** was run on client mode it simply connects to respective server and starts to sent data either in TCP or UDP as per mentioned in the command. TCP and UDP test via **iperf** to measure network performance is described in detail as following.

TCP Test
To carry out the TCP test first the **iperf** command was run in server mode in a server.

```
root@ubuntuserver:/# iperf -s
```

When this command was run the server waited for the connection and data from the clients as below.

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Then the **iperf** command was run in client side in a client mode so that the client can send the TCP data to the server.

```
root@ubuntuclient:/home/pratique1# iperf -c 192.168.0.1 -i 10
```

After running this command the througput and bandwidth of the network can be measured as shown below.

```
------------------------------------------------------------
Client connecting to 192.168.0.1, TCP port 5001
TCP window size: 86.8 KByte (default)
------------------------------------------------------------
[ 3] local 192.168.0.2 port 33064 connected with 192.168.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 3]  0.0-10.0 sec  1.09 GBytes   939 Mbits/sec
[ 3]  0.0-10.0 sec  1.09 GBytes   939 Mbits/sec
```

Actually this is just a single instance of the command to illustrate how it works. In the project a script was written to run **iperf** command numerous time and the data were pushed in the array each time command ran and the average of the data was saved in an output file. Basically to make task easier the command was run for 10 times in each iteration. Then the average of the data when the command ran for 10 times was written to output file. The similar task was repeated up to 10th iteration to recieve more data and to be assured of reliability of the data. In the TCP test the bandwidth and the throughput of the network was measured.

UDP Test
To carry out the UDP test first the **iperf** command was run in server.

```
pratique@ubuntuserver:~$ iperf -s -u
```

When this command was run the server waited for the connecton and data from the clients.

Then the **iperf** command was run in the client side in a client mode so that the client was able to send UDP data to the server.

```
root@ubuntuclient:/home/pratique1# iperf -c 192.168.0.1 -i 10 -u -b 900M
```

The successful implementation of the command gave the throughput , bandwidth , jitter and packet loss on the network as depicted below.

```
Client connecting to 192.168.0.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 33525 connected with 192.168.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   970 MBytes   814 Mbits/sec
[  3]  0.0-10.0 sec   970 MBytes   814 Mbits/sec
[  3] Sent 691784 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec   969 MBytes   813 Mbits/sec   0.018 ms  279/691783 (0.04%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

That was just a illustration of the **iperf** command for UDP test. For the experiment we need to run the command various time for precise data. Like in the TCP test the same scripts were implemented as this script was written for both TCP and UDP test. The script works exactly in the same manner like in TCP test mode. The only difference is that in UDP test , the more network metrics were measured. Precisely, bandwidth throughput, jitter and packet loss was measured in the UDP mode. The relevant data was saved to the output file for future utilization to create table and graphs.

Implementation of httperf

The **httperf** is a quite handy tool to measure the webserver performance. It generates http request load and sends to the server. The implementation of **httperf** was carried out in a follwing steps.

Firstly the **httperf** was installed in the both client and server.

Then the **httperf** command was run in the client to send the various http request.

```
httperf--hog --server 192.168.0.101 --rate 1 --num-con 2000 --num-call 1 --timeout
5
```

After running this command the required parameter like average request sent to the server, average reply sent by the server, and the error rate was collected. The time out factor was set to 5ms, it the request does not receives reply within 5ms it will be considered as error. The figure below shows the sample output of the **httperf** command.

```
httperf --hog --timeout=5 --client=0/1 --server=192.168.0.101 --port=80 --
uri=/ --rate=100 --send-buffer=4096 --recv-buffer=16384 --num-conns=200
--num-calls=1
httperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to
FD_SETSIZE
Maximum connect burst length: 1

Total: connections 200 requests 173 replies 34 test-duration 9.652 s

Connection rate: 20.7 conn/s (48.3 ms/conn, <=192 concurrent connections)
Connection time [ms]: min 898.8 avg 3263.0 max 6808.8 median 3219.5
stddev 1945.6
Connection time [ms]: connect 306.7
Connection length [replies/conn]: 1.000

Request rate: 17.9 req/s (55.8 ms/req)
Request size [B]: 66.0

Reply rate [replies/s]: min 5.0 avg 5.0 max 5.0 stddev 0.0 (1 samples)
Reply time [ms]: response 2491.6 transfer 771.2
Reply size [B]: header 200.0 content 508362.0 footer 2.0 (total 508564.0)
Reply status: 1xx=0 2xx=34 3xx=0 4xx=0 5xx=0

CPU time [s]: user 0.53 system 9.12 (user 5.5% system 94.5% total 100.0%)
Net I/O: 1750.6 KB/s (14.3*10^6 bps)

Errors: total 166 client-timo 166 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

The above explanation was just a single instance of the command and simply described how does the command operates in general. For the experiment we need to run same command for many times and need to average the data from them and again need to start next iteration with increasing the request rate. so, it is tedious task to type command every time and collect the data by hand. So, the script was written which runs the **httperf** command with desired number of times per iteration and retrieves desired parameter from output and saves to the file. On the next iteration it starts with increasing the number of request and does the same task. This task was repeatedly done until the server was saturated that means server could not send any reply . But normally the server was replying high

number of requests and it was very time consuming to saturate the server in normal condition so a php script was implemented which generated huge number of random numbers and sorts them thus giving the server heavy load. When this extra load was applied the server was saturated quite fast. The average reply rate was decreasing as per increasing the number of client sending request to the server at the same time.

3.2.2 System Hardware Implemented

CPU 2.3 core 2 duo Intel.

Hard Drive 69 GB

Physical Memory 4 GB.

Actually all the client had the hardware specification mentioned just above. The server also has the same hardware specification. The host and guest operating system implemented was Ubuntu server 12. 10 the 64 bit version. The server and the client had the same OS.

3.2.3 Virutal Server Setup

Due to unavailability of the powerful server the experimental setup and plan was changed. The all the powerful servers were allocated to other student and which were free unfortunately did not support virtualization. So, the new way of experiment was found. The one of the machine in the network which was working as a server in bare metal setup was used as a virtual server and other clients were kept in same condition as they were before in bare metal experimental setup. The virtual server setup was done in both KVM and Vmware. The following section describes the experimental setup in KVM and Vmware respectively.

Server Setup on KVM

Firstly the machine which was acting as a server for bare metal setup was chosen as for creating virtual server. Then KVM was installed. Then the next task was to create the network bridge so that the virtual machine created in the Ubuntu can be accessed by the other physical clients on the network. The following steps were followed to create network bridge.
The package bridge-utils was installed.
Then the /etc/network/interfaces file was configured to met as per requirement. The configuration file can be found in the appendix section.
Then network service need to be restarted.
Then image based VM was created in KVM as following.
A new directory for each VM that was dersired to create was created like `/var/lib/libvirt/images/vm1`
vmbuilder tool was used to create VM. **vmbuilder** uses a template to create virtual machines - this template is located in the `/etc/vmbuilder/libvirt/`

directory. Firt a copy was created as following.

```
mkdir -p /var/lib/libvirt/images/vm1/mytemplates/libvirt cp
/etc/vmbuilder/libvirt/* /var/lib/libvirt/images/vm1/mytemplates/libvirt/
```

To partition VM a file called **vmbuilder**. partition was created Where the partition were defined with their sizes.
The script `/var/lib/libvirt/images/vm1/boot.sh` was written which installs **ssh** on the vm and also sets default user and password which looks like this.

```
# This script will run the first time the virtual
machine boots
# It is run as root.

# Expire the user account
passwd -e administrator

# Install openssh-server
apt-get update
apt-get install -qqy --force-yes openssh-server
```

The **vmbuilder** was used to create the vm so the **vmbuilder** was run in the directory /var/lib/libvirt/images/vm1/ as following.

```
vmbuilder kvm ubuntu --suite=quantal --flavour=virtual --
arch=amd64 --mirror=http://de.archive.ubuntu.com/ubuntu -o
--libvirt=qemu:///system --ip=192.168.0.101 --
gw=192.168.0.1 --part=vmbuilder.partition --
templates=mytemplates --user=administrator --
name=Administrator --pass=howtoforge --addpkg=vim-nox --
addpkg=unattended-upgrades --addpkg=acpid --addpkg=linux-
image-generic --
firstboot=/var/lib/libvirt/images/vm1/boot.sh --mem=512 --
hostname=vm1 --bridge=br0
```

The disk image is located on `/var/lib/libvirt/images/vm1/Ubuntu-KVM/` path in a qcow format.
After sucessful completion of the virtual machine it was started . Then this virtual server was accessed by **ssh** and /text**iperf** and **httperf** were installed along with other necessary packages.

3.2.4 Server setup on Vmware

To implement Vmware virtualization firstly, the **Vmware Escxi 5.1.0** was installed on the physical machine.

Then the Escxi host was configured and managed via Vsphere client and ssh.
Virtual machine was created.
The ubuntu 12.10 guest was installed on the VM.
The Escxi had two NIC card one was detected by Escxi which was onboard but other NIC on the PCI slut was not detected because of no automatic driver for the NIC wphich was being used.
Then detected NIC was bridged with physical switch. So public ip was recieved by Escxi host and the virtual machine .Then this virtual machine was regarded as a virtual server for physical clients outside.With the help of second NIC the virtual machine was bridged with physical switch and thus via physical switch the other clients were brought in a same network with virtual server inside Escxi.

Implementation of iperf

The implementation of **iperf** is exaclty similar to the implementation of **iperf** in bare metal experiment. The same script which was used in bare metal experiment is used here. The only change is that in bare metal experiment the server was a physical server whereas here the server is KVM virtualized server.

Implementation of httperf

The implementation of **httperf** is completely similar with bare metal experiment except the server being the KVM virtualized VM. The same script used in the bare metal experiment was deployed here and the works exactly same as experiment in bare metal setup.

3.3 Result and Analysis

The data collected from experiment is now represented by table and graphs . These graphs will be also explained briefly with some analysis in this section. The result is described in two separate part as following.

3.3.1 Bare metal Experimental result

iperf Result

The table 3.1 represent the how datas are sampled in each test. This is just a glance of the data while sampled during expriement to illustrated how the experimental data were collected and how is the pattern. In this way the experiment was performed 10 times and each time the command was run for 10 times and average of data after running command was taken as a single data. This step is repeated for 10 times to get get 10 datas. While plotting graphs the average of these 10 datas is calculated and taken as single precise data and the graph was generated. The graphs that were generated from the **iperf** test are explained below briefly.

TCP average bandwidth (Mbits/sec)										
1 client	936.9	936.87	936.8	936.83	936.875	936.86	936.85	936.87	936.862	936.86
2 clients	472.8	472.33	472.65	472.7	472.55	472.44	472.47	472.557	472.462	466.63
3 clients	306.5	306.75	306.43	306.3	306.34	306.33	306.257	306.1	306.06	305.94
4 clients	235.6	235.65	235.6	235.6	235.62	235.6	235.6	235.612	235.622	239.19
5 clients	122.9	123.2	123.03	123.15	123.3	123.216	123.214	123.237	123.3	123.23

Table 3.1: Table for average bandwidth of TCP test.

TCP average Throughput (Gigabytes for single client and Megabytes for others)										
1 client	1.09818	1.098	1.098	1.098	1.098	1.098	1.098	1.098	1.098	1.098
2 clients	563.5	563.3	563.36	563.25	563.133	563.14	563.24	563.125	562.98	562.99
3 clients	365.3	365.6	365.16	364.97	365.06	365.1	365.014	364.85	364.81	364.66
4 clients	280.8	280.8	280.83	280.8	280.8	280.783	280.785	280.787	280.789	285.05
5 clients	146.7	146.9	146.7	146.825	147	146.9	146.95	147.022	146.94	146.914

Table 3.2: Table for average data transfer per 10 seconds of TCP test.

The table 3.2 represents the sampling of data in TCP mode. Here average TCP bandwidth is collected and it is just presented to illustrate the collection of the data and how do the data looked like. These 10 datas per each client which is itself each data being average of ten datas are averaged and a single data is taken for plotting the graph. In the same way the table below just reflect the data as a average throughput of the network. The data was sampled on the same manner as that of average TCP bandwidth.

The figure 3.3 represents the comparison between the bandwidth of the network in two different modes UDP and TCP. If we observe on the graph we can see that in TCP mode of data transfer the network has higher bandwidth than in UDP mode. The interesting thing the graph is revealing that when there is bulk of data transfer in network TCP has significantly higher bandwidth in comparison with UDP . This happens when a single client is transferring data and is able to utilize maximum bandwidth of the network. On the other hand when there is more clients simultaneously transferring data, the bandwidth was distributed among them. That is true for both TCP and UDP mode of data transfer. When there is more clients sending data at the same time the TCP mode posses slightly higher bandwidth in comparison with upd mode of transfer unlike the case of single client.

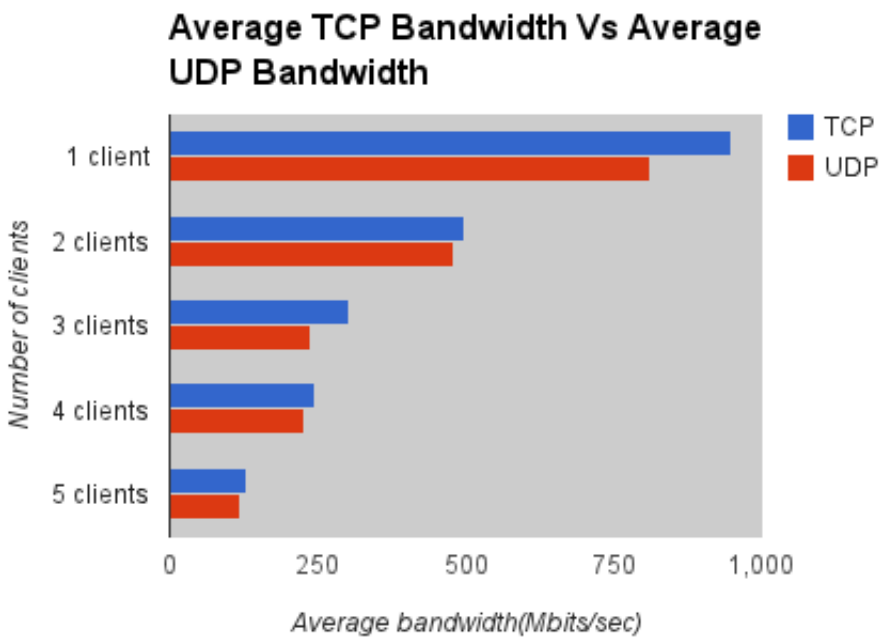


Figure 3.3: Comparison of the bandwidth of network with UDP and TCP packets.

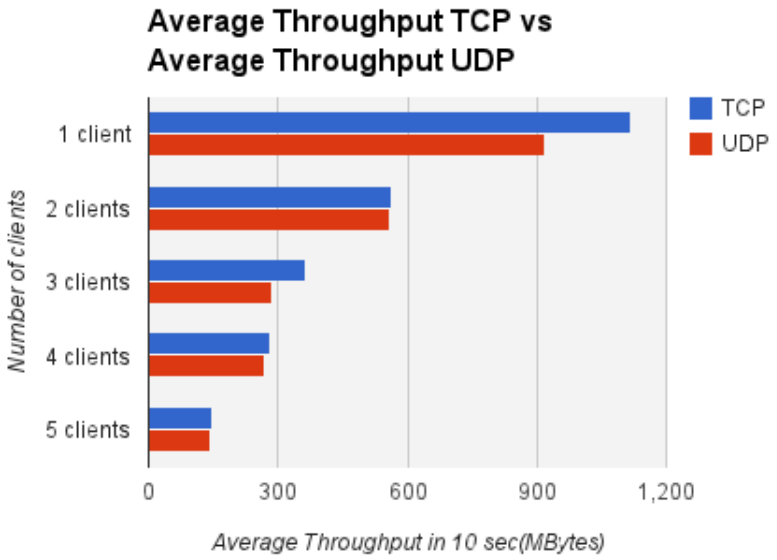


Figure 3.4: Comparison of the throughput of network with UDP and TCP packets.

Figure 3.4 represents the comparision of the througput of UDP and TCP packets in a network. To measure througput the average data transferred in 10 secs with maximum bandwidth connection was collected in both TCP and UDP mode. The figure gives information that TCP througput

is higher than the UDP throughput. When there is single client it uses maximum bandwidth and transfer maximum data. When clients are added one by one the bandwidth is distributed among clients and thus reducing the throughput. It is clearly observed in the graph the througput is much higher with only one client in the network sending packet in TCP mode than in UDP mode but after adding other clients there is slightly more throughput in TCP mode than in UDP mode data transfer.

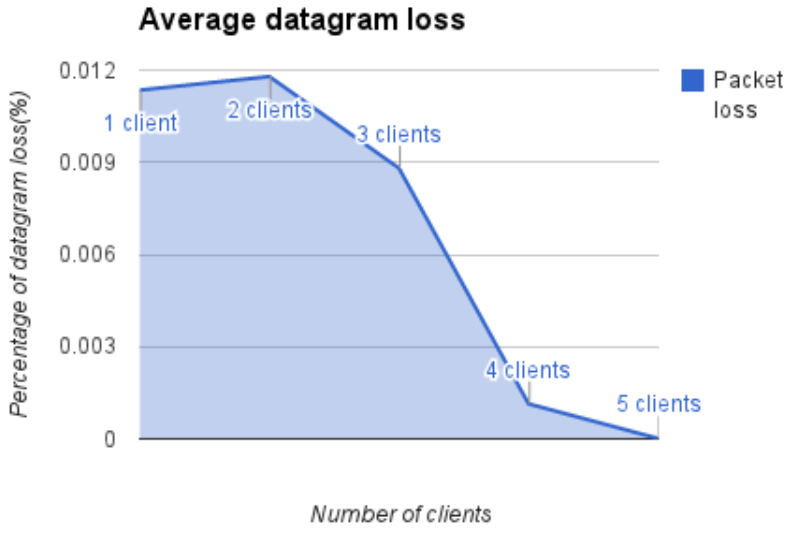


Figure 3.5: Average Datagram loss in the network in UDP test.

The figure 3.5 shows the average datagram loss while clients being sending UDP data to the server. The graph compares datagram loss among the different number of clients. The datagram loss occured on client is assed upon gradual addition of the clients on the network. At a glance on the graph, it is clearly is noticed that increasement on the active client leaded to less packet loss on the network. The pattern of the graphs reveals the information that when there is less data being transferred then there is less chance of the loss of datagram as well. With a sharp observation on the graph it is seen that when there is peresence of second client on the network the datagram loss rised slightly. After additon of third client the bandwidth and throughput is distributed among these clients and the less data is sent by each client thus leading the sharp decline on datagram loss. This trend is continued with addition of the fourth client on the network. when there was presence of fifth client on the network the datagram loss was negligible.

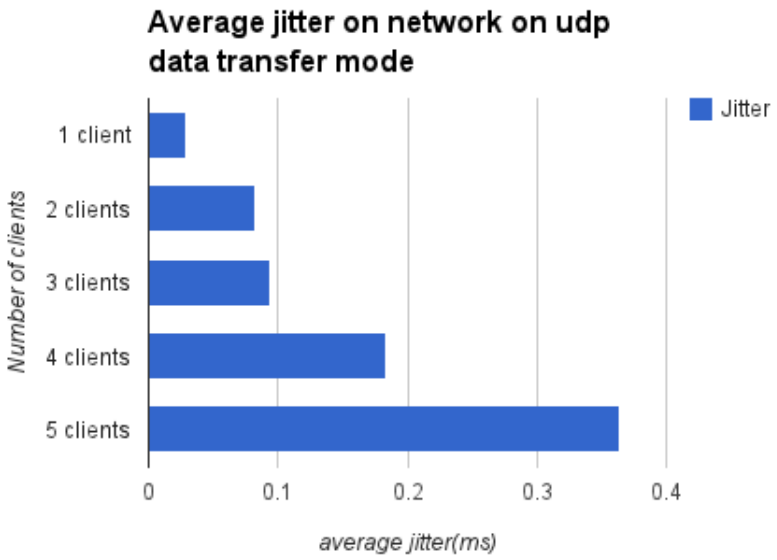


Figure 3.6: Average jitter in the network in UDP test

The figure 3. 6 gives a guick view of jitter on the network and compares the average jitter over number of clients being active on the network. On a close observation of the trend of graph, it is clearly visible that jitter on the networks is increasing when there is increase on the number of the clients on the network. When there was only one client and server active on a network it is recorded the least jitter on the network. After addition of one more client the jitter increased slightly. when the third client was added there was not significant changes on the jitter of the network. However when active participation of the fourth client brought littile bit irregularity on the network leading to the significant increase on the jitter . The same pattern was observed upon the addition of the last client and whe noticed the maximum jitter on the network. After analysis of the result it was figured out that additon of the active client causes some irregularities on data transfer and more the client on the network more strange behaviour can be expected thus it is most likely to lead high jitter when many clients simultaneously sending package on the network. I

httperf Result

For measuring web server's performance , first just **httperf** command without any extra load was implemented but it was very hard to saturate the web server. The web server was able to reply a lot of request from a client . The raw data of the various experiments were collected and average was calculated to generate graph.

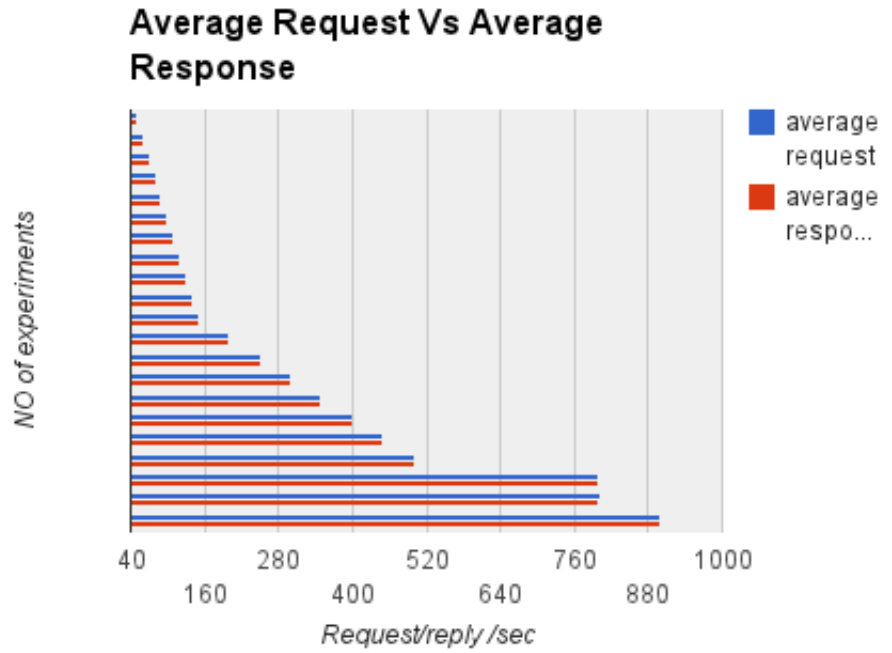


Figure 3.7: The request sent by clients and response sent back by server without any extra load.

Figure 3. 7 depicts the average request sent by a client and the average reply sent back to requests by server. Here there only one client and server. In the figure it is clearly noticed that the server is replying huge no of the requests without any error. The reply rate is equal to the request rate. This result was obtained when client was sending http request via **httperf** command without any extra load. When server was replying 800 requests per second, there was problem with client to generate more request. The situation was reserved client was unable to send more request instead of saturating the server. So, it was decided to generate extra load to server with the help of script to easily saturate the server. The script generates huge number of random numbers and sorts it so that was proven as a heavy load to server.

The figure 3.8 represents the average request sent by clients to the server before the server gets saturated. The graph shows that when there is a single client it can send more requests to the server. When many client are sending the request then the capacity of the server to reply the requests from clients gets deteriorated. Actually, it has also been observed that sometimes upon increasing the request the client is still unable to increase the no of the request. The pattern of the graph shows that when the server is down then the clients are steadily increasing the request without any reply from the server and the error is expected high in that situation. Actually in the graph only ten sample of the succesive data has been plotted. When the server is statured there is same pattern shown by every clients as we can see that the request rate is increased by client in a same rate. The figure 3.9 depicts the average responses sent by the server on the behalf of the request

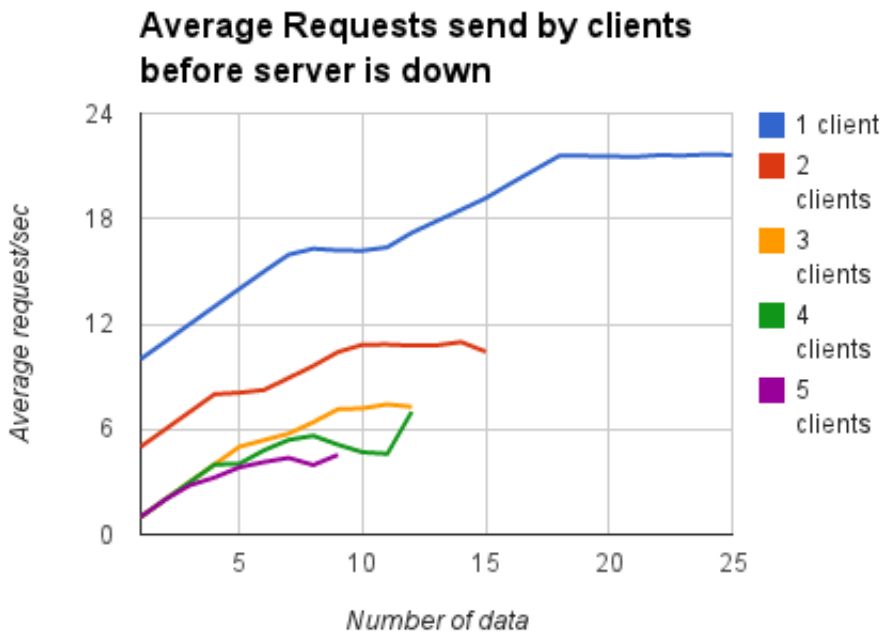


Figure 3.8: The average request sent by clients til server is saturated(with extra load).

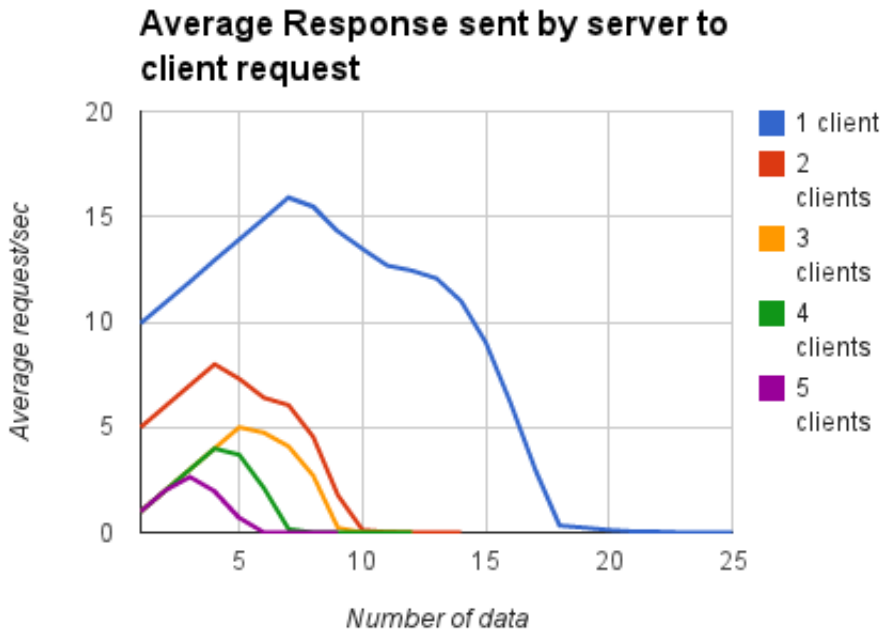


Figure 3.9: The average response sent by server til server is saturated.

from client. When there was single client the server is able to send more responses to the request from on it. Upon addition of the second client the capacity of the server is divided between them and it can reply less number of the requests. When there was only single client the server was able to handle request rate of 17 with some errors and the perfomance of the server starts to deterioarate gradually upon incresement of requests beyond that and gets totally down after rate of 20 requests per second. But when there is presence of the second client which is also sending the request at the same time then the server can handle 8 request without deterioration in capacity but after increasing more request from client the server's capacity started to degrade and it started to reply less number of request from client and was completely down when clients were sending 10 requests per second. This same pattern applies for addition of other clients as well. So, it had been noticed that the total capacity is distributed among clients and when there is addition of new client on the network the response given by server to former clients affects as it has to serve the new clients as well.

Figure 3.10: The average error occurred during course of reply sent by server to clients requests.

The figure 3.10 simply shows the average error ocurred while server is replying the request of the clients. If server is unable to reply request within 5 sec it is also regarded as error. On close observation of the pattern of the graph it is observed that initally the error is zero when the server was replying request when there was less number of request from the client. When the request was increased the server slowly started to generate error because it was not able to response all the requests from clients. Finally when the server was down the error was generated very high. It is also

observed that when there was high number of requests the number of the error was also high just before server was saturated that is why when there was single client there was high number of request and error was high when the server was just being saturated. On the other hand when there was five clients sending requests at the same time , the last client could not send so much requests and thus the error was less in a numeric value. But if we compare proportionaly between error and request then it might be different.

3.3.2 Result of virtualization Expriment

In this section the result of experiments on a virtual environment is explained in detail. The result is described in a seperate section as per virtualization technology . On the similar way the experiment is also explained on the two separate section as **iperf** result and **httperf** result. The result includes various graphs and there explanation along with comparison and analysis.

3.3.3 KVM Experimental Result

The result of experiment on KVM virtualization platform is described in detail. Basically, the result is described separately as a **httperf** result section and **iperf** result section.

iperf Result

The figure 3.11 represents the comparison of average TCP bandwidth and UDP bandwidth on KVM virtualization. It is clearly seen that bandwith for single client in TCP test is little bit higher than UDP. When more clients are present on the network sending data the bandwidth is distributed among clients. After addition of second client UDP has slightly higher bandwidth than TCP. This is because when less data is transfered UDP is better than TCP because UDP does not perform handshaking and acknowledgement.

The figure 3.12 depicts the comparison of average TCP throughput with average UDP throughput. When there is only one client on the network the TCP throughput is higher than UDP throughput in the same manner like bandwidth in figure 3.11. When there are more than one clients active on the network and sending data the throughput of UDP is slightly higher than TCP throughput. It can be predicted that the higher bandwidth led to higher throughput in UDP data transfer.

The figure 3.13 simply gives information about the average percentage of the datagram loss on UDP . If a close observation is made in the figure, it can be noticed that the datagram loss is falling down with the increasement on the number of the active clients on the network. When there is a only one client active on the network there is high datagram loss in a comparison with presence of the multiple active clients. When the number of the active

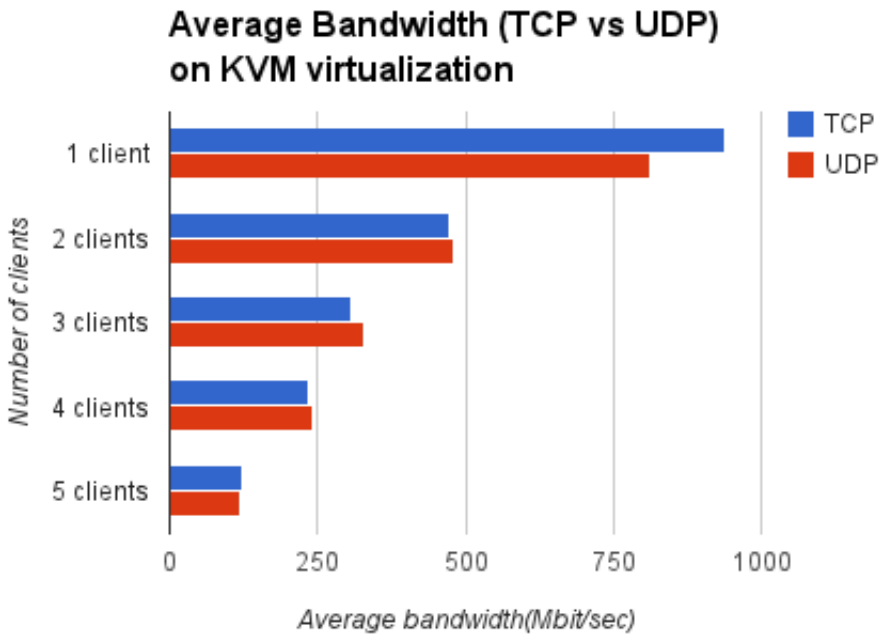


Figure 3.11: The comparison of average TCP bandwidth and UDP bandwidth on KVM .

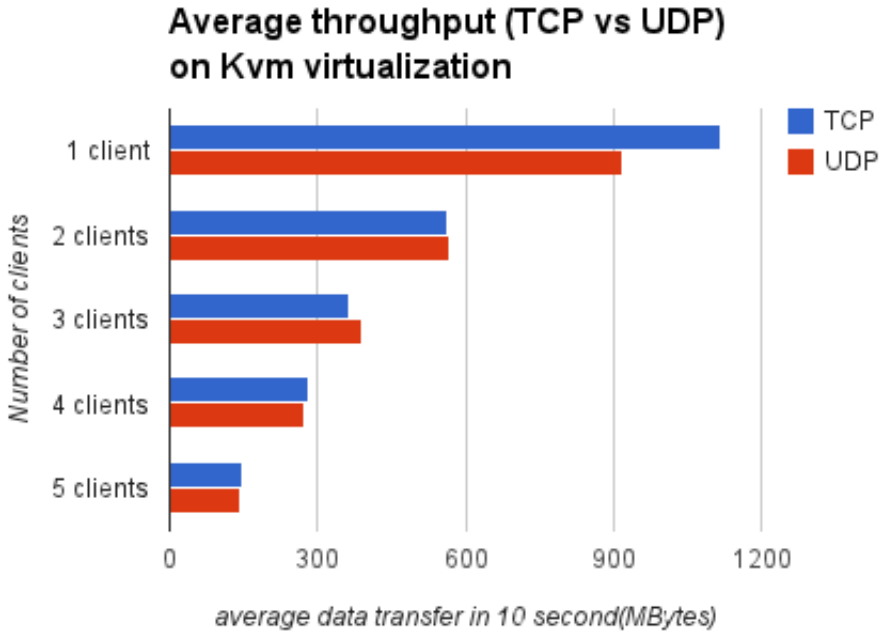


Figure 3.12: The comparison of average TCP throughput and UDP throughput on KVM.

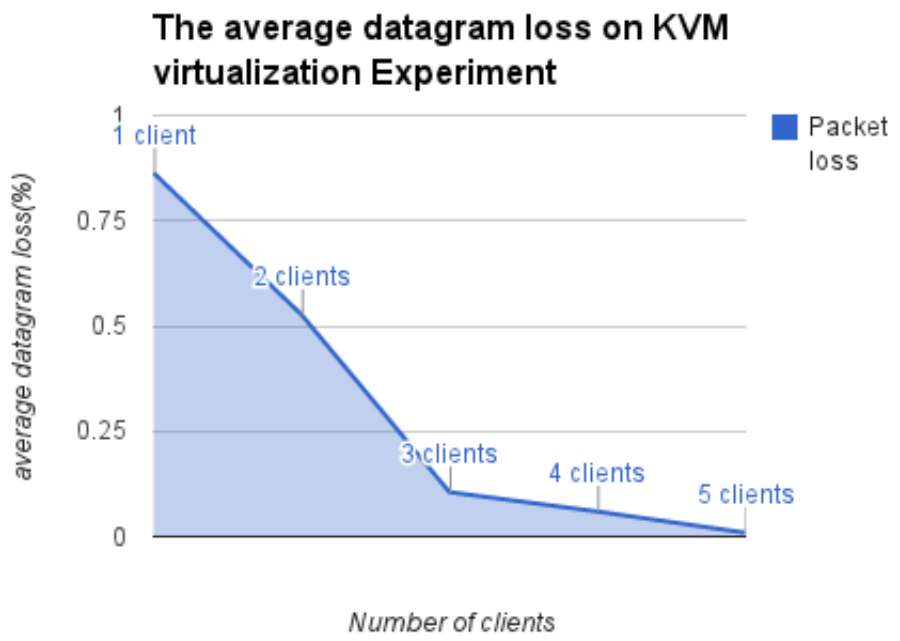


Figure 3.13: The average datagram loss on UDP data transfer on KVM.

clients increases then there is less bandwidth availabe allocated for the clients resulting less throughput. The trend of graph shows that the datagram loss is high when thre is only one client utilizing the whole bandwidth and when there is a presence of the. more clients the datagram is reudicing and for fifth clients it is almost neglible amount of datagram. When the client is sending less amount of the data then there is less datagram loss . From this result we can conclude that UDP is efficient when the clients is supposed to send less amount of data but when a bulk of data is required then TCP can be better than the UDP. The figure 3.14 depicts the average jitter recorded on the network while server is virtualized on KVM. The pattern of the graph reveals that the jitter is increasing upon addition of active clients on the network. Upon gradulal additon of the clients it is seen more the client is added more chance to experience less datagram loss.But sharp observation in the graph, the jitter for first and second client is almost same and same applies for clients 3and clients 4. The jitter is increasing slowly and reaches to peak when all the clients are active and want to send data at the same time. There is a distinct rise on the jitter when the fifth client is added on the network when four clients are functioning at the same time.

httpef Result

The httperf result described as following with a extra load being implemented on server via script like in a bare metal experiment.

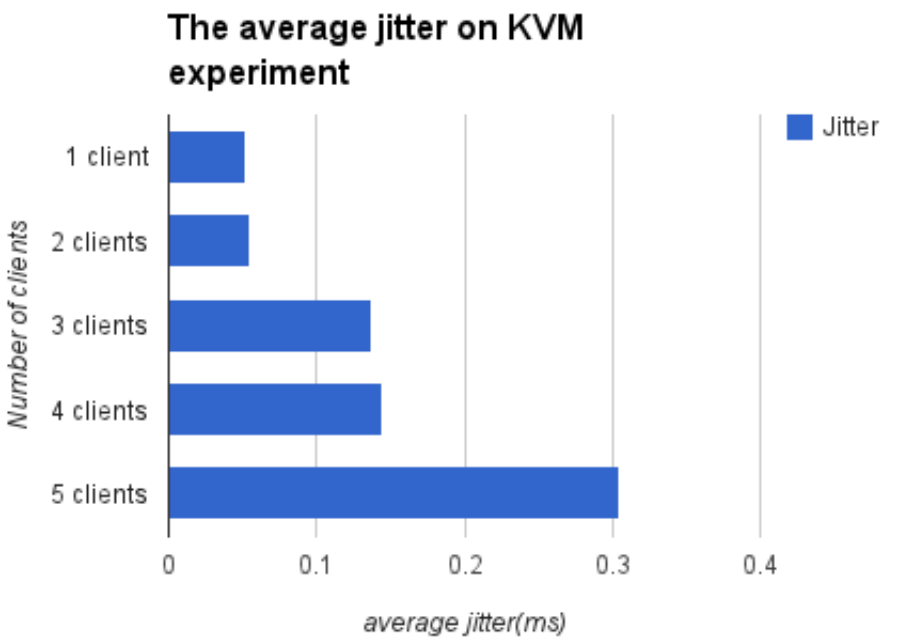


Figure 3.14: The average jitter on UDP data transfer on KVM.

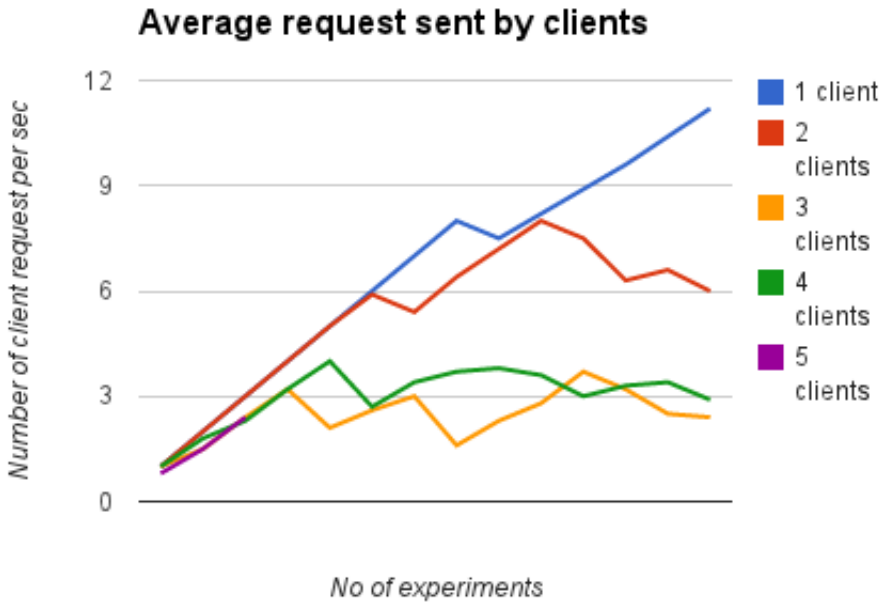


Figure 3.15: The average request sent by clients to server TCP on KVM.

The figure 3.15 represents the average number of the request sent by clients. On the sharp observation, we can see that first the graph for each graph is linear and then suddenly starts to drop and again it rises up. The

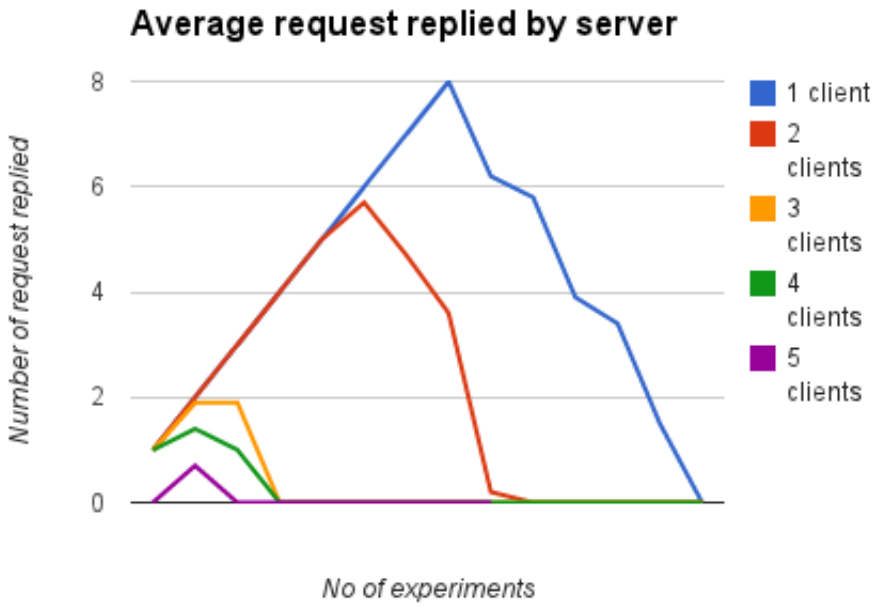


Figure 3.16: The average reply sent back by server to clients requests on KVM.

graph is linear until the server is not loaded. When the server is loaded and clients is sending request but did not get reply from server and there is sudden drop on the client request for a while when server is down completely the client still sends the request and the graph rises up again. When there is only one client sendig http request then it can send more http request before server gets saturated completely. When the more client simultaneously sending http request to the server, the capacity of clients to send request also degrades as shown in the graph. The figure 3.16 shows the average reply sent back by servr on the behalf of the clients requests. In the figure it is noticed that when there is single client the number of the request replied by server is higher then compared to when there is more clients sending requests at the same time. The trends of graphs gives information that the server replies intial request linearly and then it gets loaded when the request is constantly increasing and then the server capacity slowly decreases . After some request the server is no more able to reply any requests which the state of saturation. More is the number of active clients sending request less is the reply sent back to clients by server. The figure 3.17 represents the average error occured during the course of replying requests from the clients by server. The error may occur in two cases , when server is not capable to reply request before time out or server is not capable of the reply some requests due to overload. If the graph is observed carefully ,it can be noticed that initially error is zero when server is able to reply all the request from the client . When the server starts to get loaded it starts to generate error and gradually server goes down by the

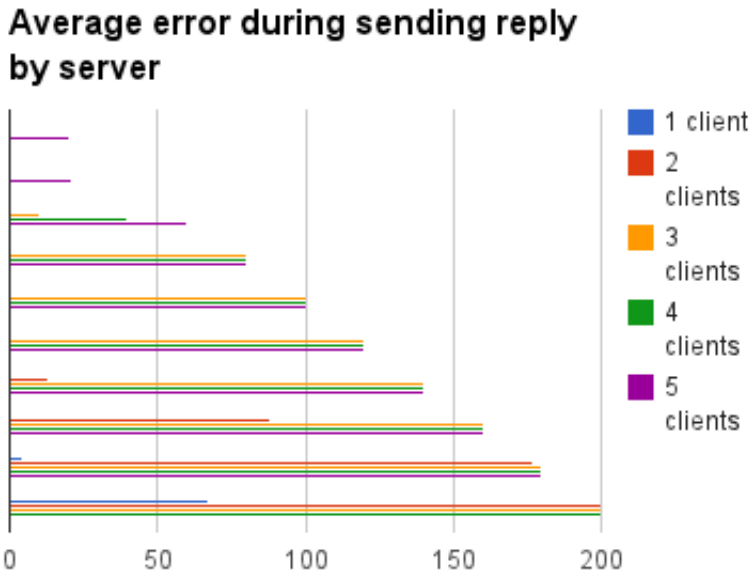


Figure 3.17: The average error occured during reply sent by server to clients request on KVM.

same time error rises gradually.When the server is saturated , it generated high error linearly. When there is only one client sending http request, the error is generated a bit late after replying maximum request .But when the server is requested by many clients at the same time it gets overloaded and starts to generate error quite early because server is not able to reply all the requests sent by the clients often it misses some of the requests a bit earlier.

3.3.4 Vmware result and comparison with others

In this the result of experiment in Vmware virtualization is explained while comparing simultaneously with KVM and bare metal result.

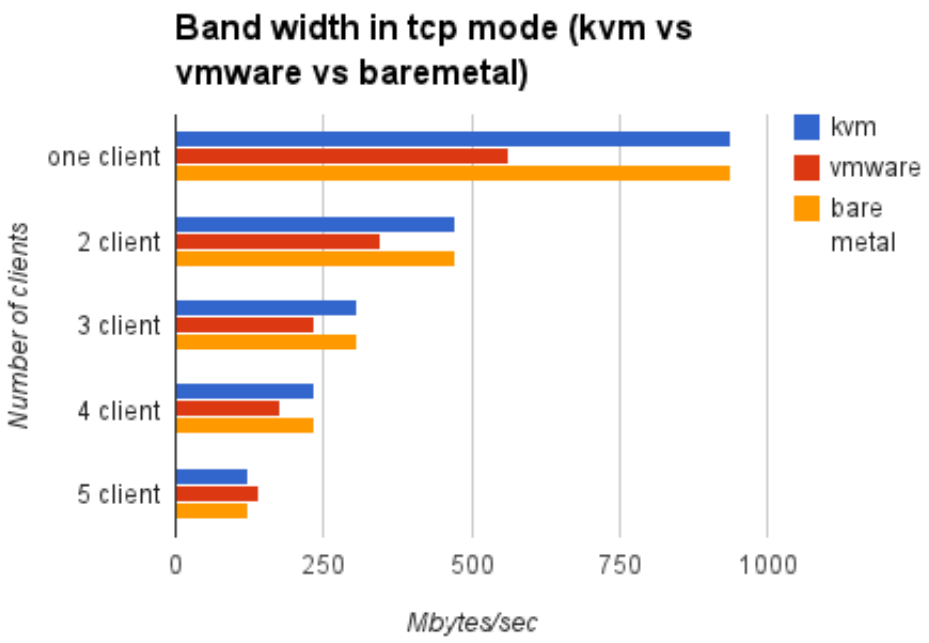


Figure 3.18: The comparison of TCP bandwidth among Bare metal, Vmware and KVM

The figure 3.18 shows the comparison of TCP bandwidth among Vmware , KVM and bare metal setup. The bandwidth is almost same for KVM and bare metal where as Vmware platform recorded a lower bandwidth in comparison with others. But there is an exception when all the 5 clients were sending data, the bandwidth in Vmware was slightly higher than others. The trend of the graph shows that when there is single client then it is connected with maximum bandwidth however when there are many clients simultaneously active the bandwidth is divided to all of them resulting less bandwidth per client on the network.

The figure 3.19 is simply a comparison of TCP throughput among bare metal, Vmware and KVM. The same trend is observed among different platform like in the figure 3.18. The KVM and bare metal perfomance level is almost same while Vmware platform showed lower throughput because of lower bandwidth.

The figure 3.20 represents the comparison of UDP bandwidth among KVM, Vmware and bare metal. Unlike in TCP mode , the Vmware platform recorded slightly higher bandwidth than others. This bit strange that it has higher bandwidth than even bare metal environment.The perfomance of KVM levels to bare metal however the client number three on bare metal showed lower value of bandwith than in other platforms . This can be regarded as a outliner. The bandwidth is maximum for single client and it is distributed to other clients when they are connected actively on the network.

The figure 3.21 depicts the comparison of UDP throughput among Vmware, KVM and bare metal platform. The trend of throughput followed the

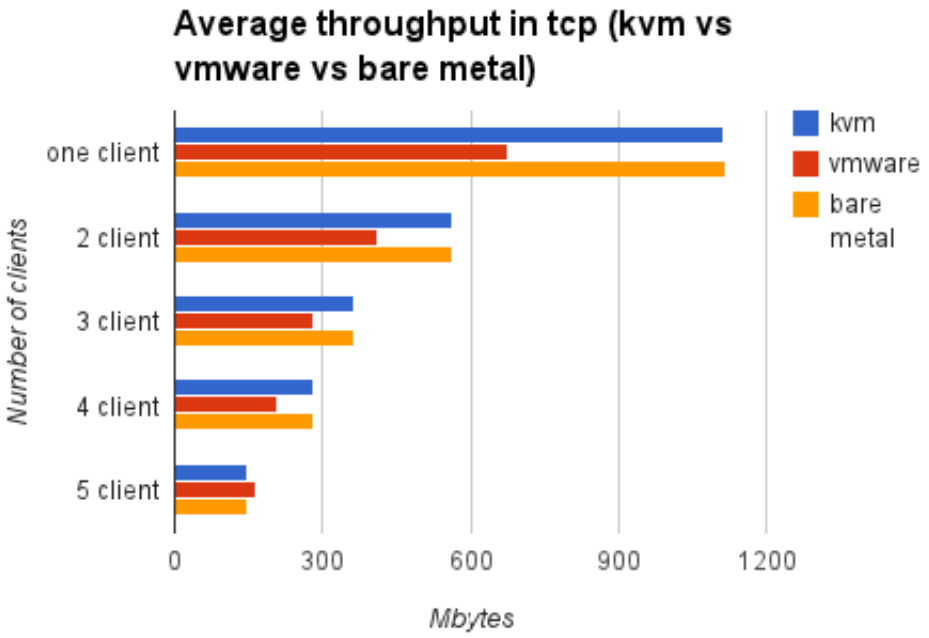


Figure 3.19: The comparison of TCP throughput among Bare metal, Vmware and KVM.

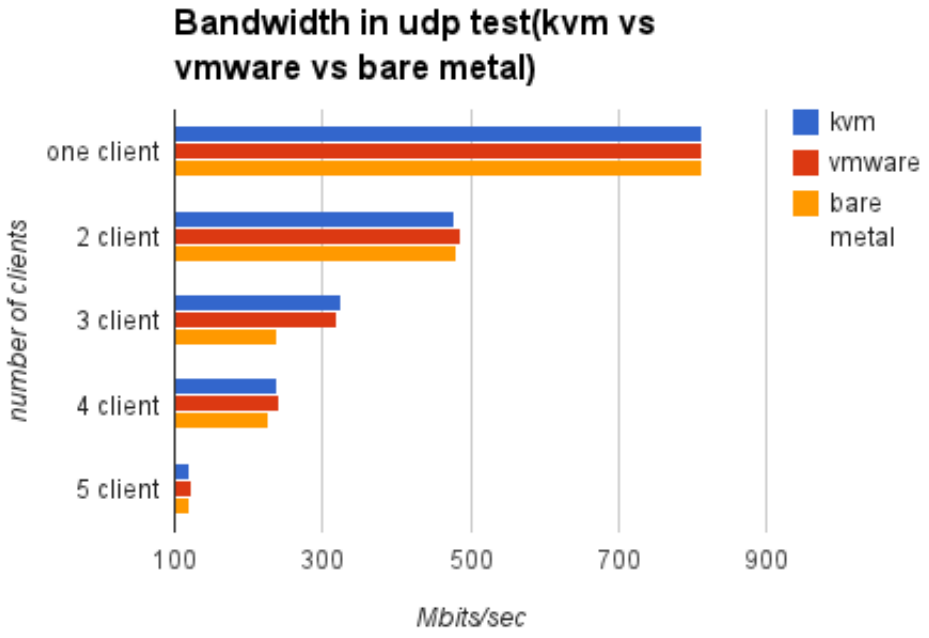


Figure 3.20: The comparison of UDP bandwidth among Bare metal, Vmware and KVM .

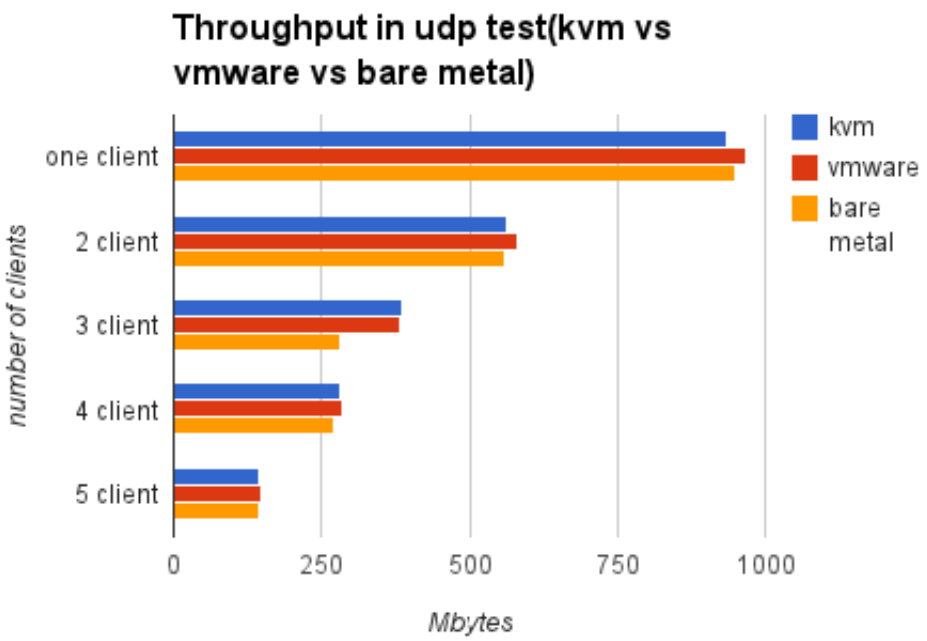


Figure 3.21: The comparison of UDP throughput among Bare metal, Vmware and KVM.

trend of the bandwidth that is why Vmware has slightly higher throughput than others.The throughput of KVM is almost same like bare metal platform. In the bare metal, client number 3 showed less bandwidth than KVM and Vmware which is a sort of exception or some abnormal behavior of client 3.

The figure 3.22 depicts the comparison of datagram loss between KVM, Vmware and bare metal platform. The most striking information of the graph is the enormous datagram loss in Vmware platform.The datagram loss is negligible for bare metal and slightly datagram loss is recorded in KVM when transfering large amount of data. The datagram loss is inversely proportional to the number of the active clients on the network.As, the number of client increases then the datagram loss is lower down .The bare metal platformance is better in the case of the datagram loss while Vmware has unreliable data transfer with even eighty percent datagram loss.

The figure 3.23 shows the comparison of jitter between Vmware, KVM and bare metal platform. The jitter on Vmware platform was much higher than others. The KVM platforms also recorded some jitter but almost negligable in comparison with Vmware. The jitter is observed very low on bare metal almost negligable. The jitter increase with increasement of number of clients on the network according to trend of the graph. In general , it is true but sometime litte bit deviation is also observed in KVM and bare metal setup.

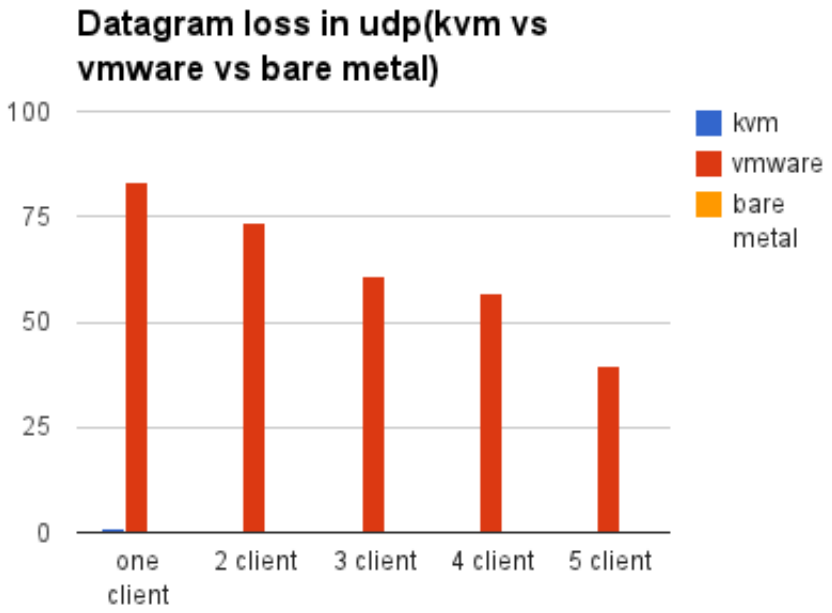


Figure 3.22: The comparison of datagram loss between Bare metal, KVM and Vmware.

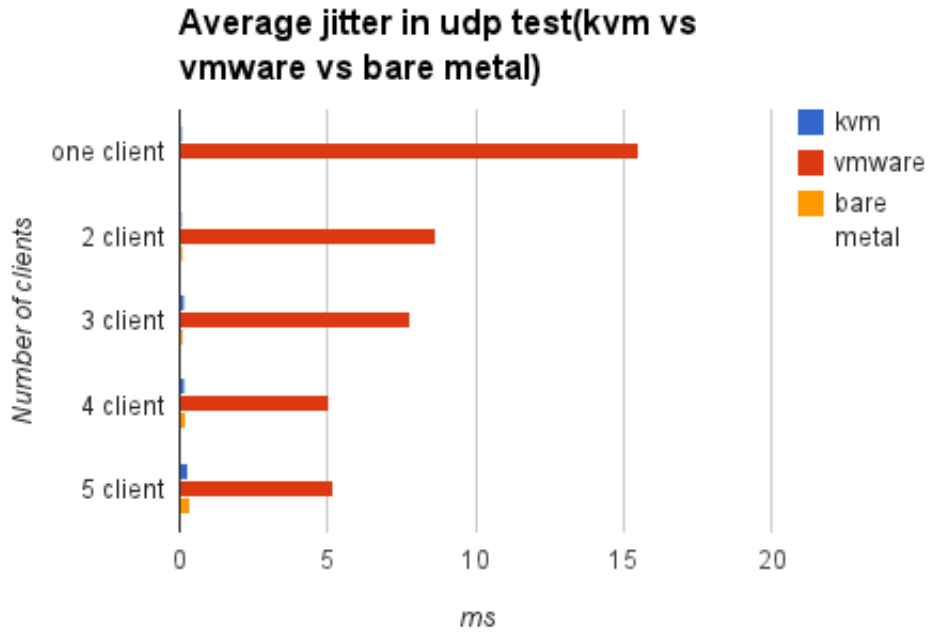


Figure 3.23: The comparison of jitter between Vmware, KVM and Bare metal.

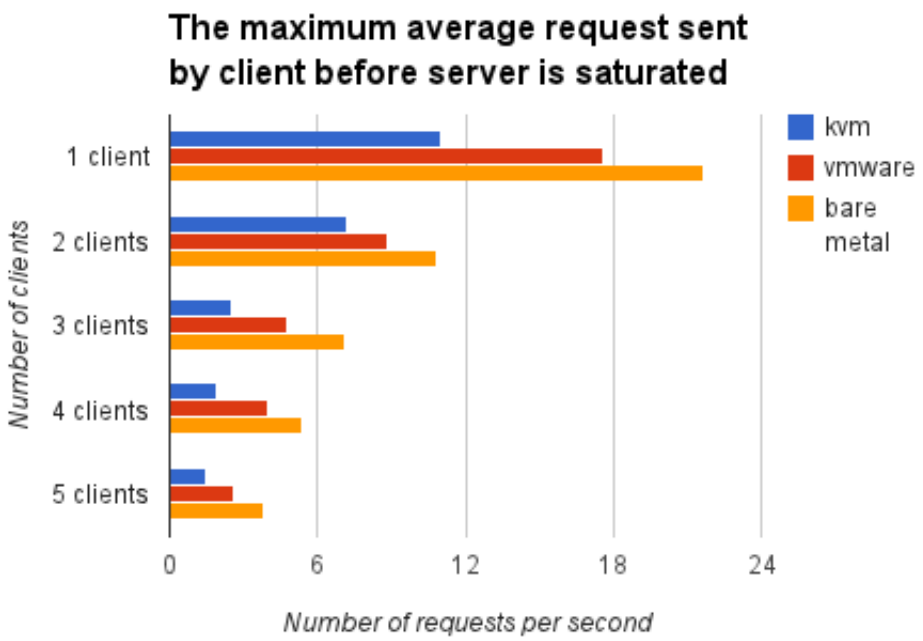


Figure 3.24: The comparison of maximum number of the requests sent by clients among Bare metal, Vmware and KVM

The figure 3.24 is graph for comparison of maximum number of request sent by client just before the server is down among Vmware, KVM and bare metal platform. It is observed that clients in bare metal are able to sent more request to server before it gets saturated. The lowest request sent by client is observed in KVM. The Vmware perfomance is slightly better than KVM. It is clearly observed from the graph that the increasement of client affects the number of request sent by client before server gets saturated. More the number of the clients less is the maximum number of request sent by clients to the server.

The figure 3.25 represents the average response sent by the server on the behalf of the clients requests on Vmware. On a close observation in the graph it is noticed that firstly the response is linear because sending reply to all requests from the clients then slowly there fall in reply and this continues until the reply is zero. When there are many clients sending request simultaneously the server is loaded and can not reply all the request from clients it does partially reply. Then when there are many more requests continuously increasing then server is beyond its capacity and is saturated at some point. To compare this result with KVM and bare metal setup it is necessary to compare the figure with figure 3.16 and 3.9. So, from the comparison between these figures it is noticed that bare metal clients got more response than others. The Vmware setup has better performance than KVM in this context.

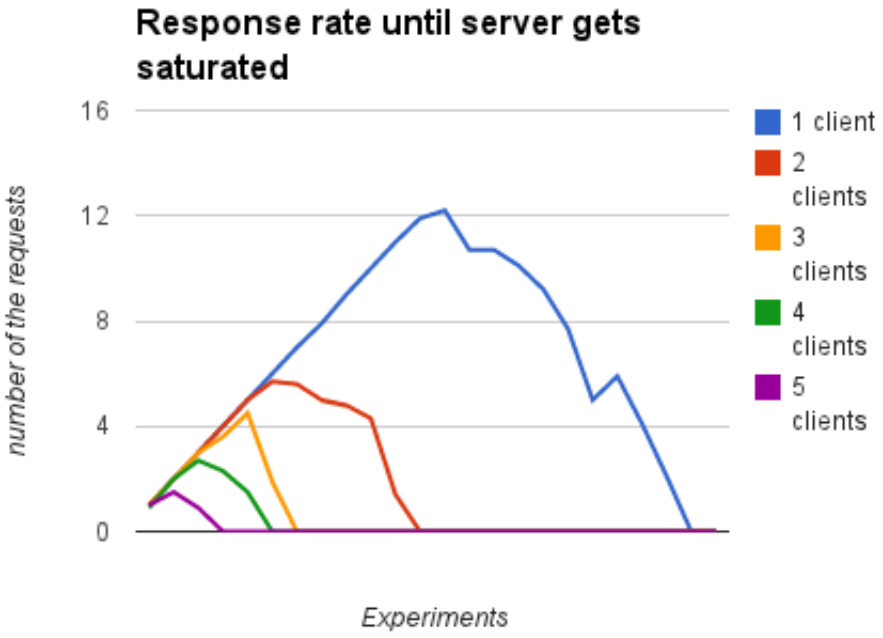


Figure 3.25: The average response sent by server to the requests from clients on Vmware.

Chapter 4

Discussion

This chapter enlights the finding and outcome of the thesis work, difficulties and challenges experienced, troubleshootings and vivid scenarios. Basically, this thesis was done on pretty familiar topic and of course many research has been done on the areas of the networking and virtualization before. The main goal of the project was to implement the virtualization technologies and find out the impact on the performance of the network when the physical network is virtualized in a virtual platform. There are various parameters which measure the performance of the network . It would have been better to cover the as many network performance as possible because in that case it can give a nice feedback about performance of the network regarding various scenarios. It was not possible to cover so many network parameters due to limitation of the time frame. So, some of the important parameters that assess the performance of the computer was measured with appropirate tool. Although it is not possible to cover all the metrics, it would have been far better if important parameters like latency, CPU overhead , memory usage had been covered. However the most significant network perfomance measuring metrics precisely naming bandwidth, throughput, packet loss and jitter was measured. The maximum bandwidth and the maximum throughput with which client can transfer data on a computer network was measured in both UDP and TCP data transfer mode and comparision is accomplished as well. The experiment was performed in twofold task. Firstly, the expriment was conducted on physical network and then result was recorded, then same experiment was carried out on virtual platform that means sever was virtualized using virtualization technologies. The main purpose of conducting the experiment on two different platforms is to make task easy to find impact of virtualization on performance of the computer by comparing the results between bare metal platform and virtualization platform. As per expectation, it was possible to figure out some impact of virtualization and find some degradation of the performance of the network when server being virtualized.

The experiment was carried out on many rounds to catch sharp transition on the experiment and gather more precise data. More precisely, the computer network was established on gradual additon of clients one after an-

other. To make more clear, first single client is connected and experiment is conducted and result is recorded. After successful completion of experiment the second client is added , same experiment is carried out and process goes so on until final client is added. This same procedure is implemented on virtualization experiment as well. That's why the comparison was convenient and making conclusion became simple and easy. However, the experimental set up was handy to compare results between two platforms, the task was tedious and very much time consuming. The experimental set up sounds easy but was time consumig because everything needed to be installed on many clients. For example, installing operating system and configuring everything took long time than it was expected.

This project is quite simple and major task accomplished is measurement of network performance on basis of network parameters like maximum bandwidth, maximum throughput , packet loss and jitter. The main emphasis was given to virtualization but bare metal part was also prominent . Besides measuring the network parameters , in this project measurement of web server performance was also carried out which also plays significant role in overall thesis project. Actually, network metrics were measured only taking account to clients which means performance of the client in the network was emphasized rather than server. Moreover, it was observed that at what speed the clients can send data to server, what the impact on the performance of the client was when there was increasement on the number of the clients on network. To observe this effect client were added manually on the network and network performance parameters were measured. That's why the webserver performance measure makes sense to measure the capacity of the server on replying http requests from client on different scenarios. The web server performance was measured based on the average response sent back by the server to the request from the clients and of course with varying number of clients on the network. On the other hand, the average error occured during reply from server to the clients requests was also measured to figure out more on measuring performance of the server.

This project might be little bit helpful to one who is interested on networking and virtualization. Basically, the content of the project are very basic. So it can be possible to get some basic ideas about how to measure network perfomance and what can be effect of virtualization on it. It also gives basic idea about measuring the performance of the web server so anyone who wants to measure the performance of the web server can get some ideas from this project . In addition, the impact of the virtualization on the performance of web server can also be visualized with this project. It also gives some ideas about the virtualization to the reader especially about KVM virtualization and Vmware virtualization technologies. It also gives basic guidelines about virtual networking which means how to setup network and experiment with virtual machine and physical client on KVM and Vmware.

The major finding of this project was measurement of the simple network metrics in order to find out the performance of the small computer network. The behavior of the clients on the network while sending data in UDP and TCP modes was also studied. The comparison of the performance of the network between bare metal environment and virtual environment has been done to some extent. Implementation of the two most popular virtualization technologies i.e. KVM and Vmware, had been facilitated and also been compared to appropirate extent. Besides this, the measurement of the web server was also carried out which is an imminent part of this project indeed.

There were not such a big problem faced during project but was passed over many small difficulties. One of big mistakes commited during master thesis was inappropirate time planning and scheduling tasks. Firstly, the research were done slightly and then topic was selected. Then hardware for project was asked with university when it was time to do experiment. Because of late demand of hardware, namely powerful server, it was not possible to get powerful server as all servers were reserved by other students and only old servers was available which rarely support virtualization. Due to this reason, there was big change on the experimental setup and plan. It was desired to move server and client into a powerful server to conduct the experiment but due to unavailibility of powerful server, it was not possible to virtualize everything . However the server was run on the virtualization platform but has same configuration like clients. So, to figure out this idea and continue in thesis, it took long time and that was simply overhead on the way of project. There is always old saying, learn from the mistakes. So, it is learnt that firstly, the experimental setup should be prepared and then other things need to be done afterwards. The main problem in the experiment was that abnormal behavior of the clients . Some of the clients gave some abnormal data. For an illustration, there is two figures represented below as following.

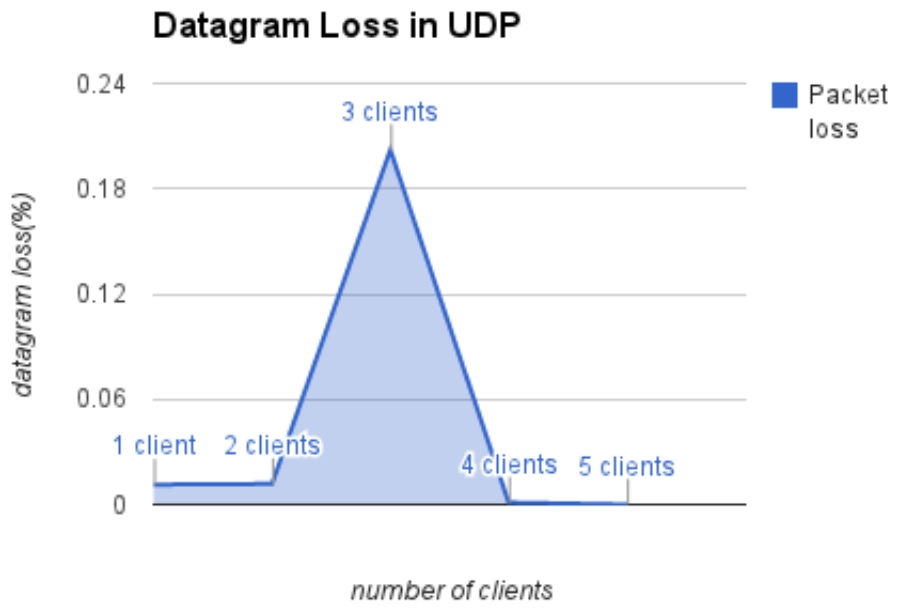


Figure 4.1: Datagram loss in UDP test

In the figure 4.1 , it is clearly seen that the irregularity shown by third client. The datagram loss for third client is extremely high in comparison with others and after that there is normal pattern shown by the other clients.

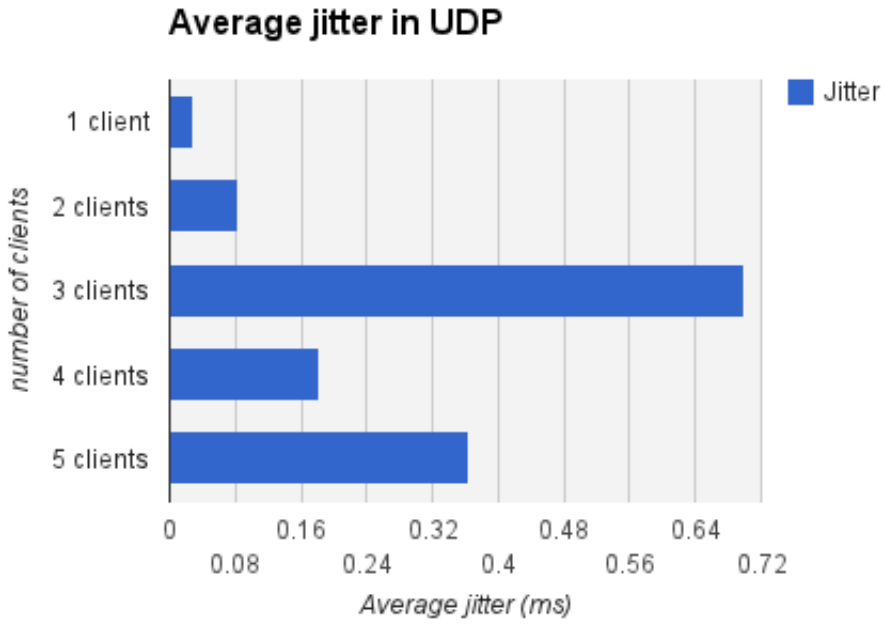


Figure 4.2: Jitter in UDP test

In the figure 4.2 , it is sharply noticed that the third client posses maximum jitter in a abnomal way just voilating the pattern shown by other clients. This sort of strange behavior was experienced many times and many odd results were observed because of that many experiments had to be conducted for number of times. Of course, this was a time consuming task. Besides that, there was little bit complication while setting up a virtual machine and physical clients in a same network and while making clients reachable to internet and configure the interfaces. In KVM, the task was little bit easy but in Vmware, it was little bit more time consuming. The NIC card in the PCI slot was not detected and it was needed to upload and to install driver manually. Briding the virtual switch with physical switch was littile bit extra work.

Nothing is complete and perfect in this world and there is no limit for making any thing better than before. The are many things which can be done to improvise this project and make it better to meet the real world problems. Actually, there are several things which can be done in the future as an extension of this project. To study and observe better impact of virtualization, the experiment can be carried out using one more virtualization technology like Xen and Virtualbox besides KVM and Vmware. It is also possible to build up clients with different operating systems like windows and mac besides linux clients. The experiment has been carried out with virtual server and physical clients so it would be nice if the clients are virtualized and the server remains the physical machine . Then it makes more idea about the impact of virtualization when there is comparison of the result between the network with physical host and virtualized server, and virtualized clients and physical server. In this project, mostly, the behavior of the client is empahsized so it is possible to emphasize two way communications between server and clients. Last but not the least, in this project the effect of switch on the performance of the network has not been studied, so it would be more realistic if the effect of switch is also taken into consideration and find out the more accurate result.

Chapter 5

Conclusion

This thesis comprises of two-fold task. Firstly, the experiment was carried out on a physical machine then the experiment was conducted on a virtual environment. The main outcome of this project is the study of the influence of the virtualization on the performance of the computer network. To facilitate this, the experiment was carried out in two phases. The idea behind such experimental setup is to observe the performance of the network in a normal condition. After a clear vision of the real performance level, it is much more convenient to figure out influence of virtualized server on the network. The measurement of performance of web server is a supplementary task accomplished in this project.
To measure the performance of the computer network, a simple and very handy tool called **iperf** was implemented. With the assistance of **iperf** the maximum bandwidth with which the clients were connected on the network and maximum throughput was measured in the TCP data transfer. On the other hand, jitter and datagram loss on the network along with maximum bandwidth and throughput was measured in UDP data transfer mode. Meanwhile, the respective data were compared as well. To virtualize a server KVM and Vmware Escxi was deployed. In fact, only the server was virtualized while clients remained the same physical machines. The five clients and a single virtualized server brought togther to build up a small private network . After experiment, it was observed that the performance was very close to the level of bare metal setup. However, on Vmware experimental setup, some deviation was noticed. Especially, abnormal datagram loss occured which signified that data transfer was not reliable on the network in UDP mode on Vmware. Jitter which signifies irregularities on the network was also found higher on Vmware in comparison to KVM. Similarly, low bandwidth and low throughput were recorded on Vmware during TCP data transfer. The datagram loss and jitter was higher in virtual environment whereas in bare metal platform it was negligible.Other results were some how close to same level.
The performance of the web server was measured with deployment of **httperf**.To measure the performance of webserver, the response sent back by web server on the behalf of the request sent was kept on the account. The performance of the web server degraded on the virtual platform. However,

Vmware had slightly better result over KVM.
In this manner, thesis is completed with some limitation and errors. The experiment has been accomplished as far as possible to meet the planning. This thesis is of course a small one but it can provide some basics information about impact of virtualization on a small computer network.

Bibliography

[1] Sahoo, J. ; Mohapatra, S. ; Lath, R. , "Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues" , *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pp. 222-226, 23-25 April 2010

[2] Amit Sing *An Intorduction to the Virtualization* Jan 2004

[3] Yunfa Li; Wanqing Li; Congfeng Jiang, "A Survey of Virtual Machine:Current Technology and Future Trends", *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, vol. , no. , pp. 332, 336, 29-31 July 2010

[4] Naveed Yaqub, "Comparison of Virtualization Technologies Overhead: Vmware and Red Hat", University of Oslo , May 2012

[5] Southern, Gabriel. "Symmetric multiprocessing virtualization. " Diss. 2008.

[6] ERP Goldberg . "Survey of Virtual Machine Research" *Honeywell Information Systems and Harvard University* , June 1974

[7] Efrem G. Mallach. "ON THE RELATIONSHIP BETWEEN VIRTUAL MACHINES AND EMULATORS" *Proceedings of the workshop on virtual computer systems* Pages 117—126, 1973

[8] Jeff Daniels, "Server Virtualization Architecture and Implementation", *Crossroad magazine* , Volume 16 Issue 1, September 2009 Pages 8-

[9] Jeremy Sugerman, Ganesh Venkitachalam, and Beng-Hong Lim "Virtualizing I/O Devices on Vmware Workstations Hosted Virtual Machine Monitor" *Vmware, Inc.* Jun 24, 2001

[10] Tal Garfinkel, Mendel Rosenblum. "A Virtual Machine Introspection Based Architecture for Intrusion Detection" *Computer Science Department, Stanford University* , 2003

[11] Arpaci-Dusseau and Remzi H. Arpaci-Dusseau "VIrtual Machine Monitor"

[12] John Scott Robin "Analysis of the Intel Pentium's Ability to Support a. Secure Virtual Machine Monitor" *USENIX Security '00* 2000

[13] Michael Fenn, Michael A. Murphy, Jim Martin, and Sebastien Goasguen. "An Evaluation of KVM for Use in Cloud Computing" *Clemenson University* 2012

[14] Yamamoto, V. Yoshihiko Oguchi V. Tetsu, "Server virtualization technology and its latest trends" , *Fujitsu Sci. Tech*, J 44. 1 (2008): 46-52.

[15] Nussbaum, Lucas and Anhalt, Fabienne and Mornard, Olivier and Gelas, Jean-Patrick "Linux-based virtualization for HPC clusters" *Montreal Linux Symposium* Mpntreal , 2009

[16] Sanghyun Han; HyunWook Jin, "Full virtualization based ARINC 653 partitioning, *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th* vol. , no. , pp. 7E1-1, 7E1-11, 16-20 Oct. 2011 doi: 10. 1109/DASC. 2011. 6096132

[17] Diane Barrett , "Virtualization and Forensics", Syngress; 1 edition (June 1, 2010)

[18] oshua White, "A survey of Virtualization Technologies with Performance Testing" Oct 15, 2010

[19] Bernard Golden, Clark Scheffy . "Virtualization For Dummies Sun and AMD Special Edition" *Wiley Publishing, Inc.* 2008

[20] Mallikarjun B Chadalpaka(January 18, 2005). "System and Method for Storage Virtualization" U. S. pat. 6, 845, 403B2

[21] Andrea Chierici , Riccardo Veraldi "A quantitative comparison between xen and KVM", *17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09)* , IOP Publishing Journal of Physics: Conference Series 219 (2010) 042005 doi:10. 1088/1742-6596/219/4/042005

[22] Y. Goto, "Kernel-based Virtual Machine Technology" *FUJITSU Sci. Tech. J.* , Vol. 47, No. 3 (July 2011)

[23] A. Binu and G. Santhosh Kumar "Virtualization Techniques: A Methodical Review of XEN and KVM", *Department of Computer Science, Cochin University of Science and Technology, Cochin, India* 2011

[24] ABBAS ASOSHEH, MOHAMMAD HOSSEIN DANESH, " Comparison of OS Level and Hypervisor Server Virtualization". *8th WSEAS International Conference on SYSTEMS THEORY and SCIENTIFIC COMPUTATION (ISTASC'08)*, Rhodes, Greece, August 20—22, 2008

[25] Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, Kang G. Shin , "Performance Evaluation of Virtualization Technologies for Server Consolidation" , *Hewlett-Packard Development Company, L. P.* , 2007

[26] David E. Williams, Juan Garda "Virtualization with Xen", SYNGRESS publication, 2007

[27] "AMD White Paper: Virtualizing Server Workloads" , *2008 Advanced Micro Devices, Inc. .* , 2008

[28] Christine, Leja, CCP, Chair Richard C. Barnier Charles L. Brown, CCP Paul , F. Dittmann Paul , Koziel Mark Welle , J. T. Westermeier, JD, CCP "Virtualization and Its Benefits" , *AITP Research and Strategy Advisory Group*, October 14, 2008

[29] "Virtualization Best Practices" , *supportconnect Ca*, May 2, 2008

[30] Andreas Hanemann, Athanassios Liakopoulos, Maurizio Molina, D. Martin Swany "A Study on Network Performance Metrics and their Composition", *Emerald Group Publishing Limited*, 2006

[31] Hyo-Jin Lee, Myung-Sup Kim, James W. Hong, Gil-Haeng Lee "QoS Parameters to Network Performance Metrics Mapping for SLA Monitoring", *Asia-Pacific Network Operations and Management Symposium - APNOMS* , 2002

[32] Masaaki Nishikiori"Server Virtualization with Vmware vSphere 4", *FUJITSU Sci. Tech. J, Vol.47, No.3, pp.356—361* , 2011 July

[33] Susanta Nanda Tzi-cker Chiueh"A Survey on Virtualization Technologies", *Department of Computer ScienceSUNY at Stony Broo* , 2010

[34] Robert Rose"Survey of System Virtualization Techniques", March 8, 2004

Appendix A

List of scripts

A.1 iperf Script

The script below was used for iperf test.

```
#!/usr/bin/perl -w
use Getopt::Std;
use strict "vars";
use Statistics::Descriptive;
my $VERBOSE =0;
my $DEBUG = 0;
my $opt_string = 'vdhtui:s:r:';
getopts("$opt_string",\my %opt) or usage() and exit 1;
if (opt{'h'}){usage(); exit 0;}
# Handle other user input
$VERBOSE = 1 if $opt{'v'};
$DEBUG = 1 if $opt{'d'};

verbose("Verbose is enabled\n");
debug("Debug is enabled\n");

# Required parameters
my $interval_rate=$opt{'r'};
my $server=$opt{'s'};
my $iteration=$opt{'i'};

debug("interval  $interval_rate \n");
if( $opt{'t'})
   {
    tcp_test($server,$iteration,$interval_rate);
   }
if( $opt{'u'})
   {
udp_test($server,$iteration,$interval_rate);
}
sub tcp_test{
open(OUT, ">tcpresult.log") or die "Cannot open file for writing";
print OUT "Transfer\tBandwidth\n";
my $server = $_[0];
my $interval = $_[2];
my $total = $interval*10;
my $command="iperf -c $server -i $interval -t $total";
debug("command $command \n");
my $limit =$_[1];
my %DATA;
for(my $i=1;$i<=$limit;$i++){
verbose("Executing $command for $i times \n");
open (IN, "$command |") or die "Cannot execute iperf\n";
# Executing the command and logging the interested outputs
my $count=0;
```

```
while(<IN>) {
if (/\s+(\d+\..*)\s+sec\s+(.*)\s+([K|M|G]Bytes)\s+(\d+)\s+(.*)/) {
if($count<10){ # print OUT $1."\t";
 debug("time interval $1\n");
 #print OUT $1."\t";
# push(@{$DATA{' Interval'}},$1);
 debug("transfer $2 $3\n");
 #print OUT $2."\t";
 push(@{$DATA{'Transfer'}},$2);
 debug("bandwidth  $4 $5\n");
 push(@{$DATA{'Bandwidth'}},$4);
}
$count=$count+1;
  }
}

my %MEAN_DATA;
foreach my $key (keys %DATA){
                my $stat=Statistics::Descriptive::Full->new();
                $stat->add_data(@{$DATA{$key}});
                $MEAN_DATA{$key}=$stat->mean();
              }

          foreach my $key (keys %MEAN_DATA){
              debug("the key used for mean is $key \n");
               print OUT $MEAN_DATA{$key}."\t\t";
                   }
          print OUT "\n";
close(IN);
}
close(OUT);
}
sub udp_test{
open(OUT, ">udpresult.log") or die "Cannot open file for writing";
print OUT "Bandwidth\t\tjitter\t\tpacket loss\t\tTransfer\n";
my $server = $_[0];
my $interval = $_[2];
my $total = $interval*10;
my $command="iperf -c $server -i $interval -t $total -u -b 1000M";
debug("command $command \n");
my $limit =$_[1];
my $avg_transfer=0;
my $avg_bw=0;
my %DATA;
for(my $i=1;$i<=$limit;$i++){
```

```
verbose("Executing $command for $i times \n");
open (IN, "$command |") or die "Cannot execute iperf\n";
# Executing the command and logging the interested outputs
my $count=0;
while(<IN>) {
print $_;
if($count < 10){
 if
(/\s+(\d+\..*)\s+sec\s+(.*)\s+([K|M|G]Bytes)\s+(\d+)\s+Mbits\/
sec(.*)/) {
 debug("time interval $1\n");
 debug("transfer $2 $3\n");
push(@{$DATA{'Transfer'}},$2);
 debug("bandwidth  $4 \n");
push(@{$DATA{'Bandwidth'}},$4);
  $count=$count+1;
  }
}
print "count= $count \n";
if (/\s+(\d+\.\d+)\s+ms\s+/) {
debug("jitter $1\n");
push(@{$DATA{'jitter'}},$1);
}
if(/\s+(\d+\.\d+)\s+ms\s+\d+\/\d+\s+\(([\d+\.\d+)\%\)/){
debug("packet_loss_percentage  $2\n");
push(@{$DATA{'percentage_packet_loss'}},$2);
}
else
{
my $packet_loss=0;
push(@{$DATA{'percentage_packet_loss'}},$packet_loss);

}


}
my %MEAN_DATA;

foreach my $key (keys %DATA){
                my $stat=Statistics::Descriptive::Full->new();
                $stat->add_data(@{$DATA{$key}});
                $MEAN_DATA{$key}=$stat->mean();

            }
```

```
        foreach my $key (keys %MEAN_DATA){
            print OUT $MEAN_DATA{$key}."\t\t";
                }
        print OUT "\n";

close(IN);
}
close(OUT);
}
sub usage{
            # prints the correct use of this script
              print "Usage:\n";
              print "-h Usage \n";
              print "-v Verbose\n";
              print "-d Debug\n";

              print "./script [-d][-v][-h]\n";
        }


sub verbose{
            print "VERBOSE:".$_[0] if $VERBOSE;
        }


sub debug{
            print "DEBUG:".$_[0] if $DEBUG;
        }
```

A.2 httperf Script

The script below was used for httperf test.

```
#!/usr/bin/perl -w

use Getopt::Std;
use Statistics::Descriptive;
use Data::Dumper;
use strict "vars";
my  $count;

# Global variables
my $VERBOSE = 0;
my $DEBUG = 0;


my $opt_string = 'vdhl:u:s:i:f:';
getopts("$opt_string",\my %opt) or usage()and exit 1;


# Print help message in -h is invoked
if($opt{'h'}){
            usage();
            exit 0;
          }

# Handle other user input
$VERBOSE = 1 if $opt{'v'};
$DEBUG = 1 if $opt{'d'};
verbose("Verbose is enabled\n");
debug("Debug is enabled\n");

# Required parameters
my $rate=1;
my $low=$opt{'l'};
my $high=$opt{'u'};
my $hostname=$opt{'s'};
my $iteration=$opt{'i'};
my $outputfilePath=$opt{'f'};



$count=$low;
my $times=0;
open(OUT, ">$outputfilePath") or die "Cannot open file for
writing";
print OUT "avg_req\tavg_res\tavg_error\n";
while($count <= $high){
```

```
        my $numberofcon=$count*20;
        my $command="httperf --hog --server $hostname --rate
$count --num-con $numberofcon  --num-call 1 --timeout 5";
        my $limit =$iteration;
        my %DATA;
        my %MEAN_DATA;
        for(my $i=1;$i<=$limit;$i++){
                    verbose("Executing $command for $i times
\n");
                    open (IN,  "$command |") or die "Cannot
execute httperf\n";
                # Executing the command and logging the
interested outputs
                while(<IN>) {
                        if (/^Request rate: (\d+\.\d)/) {
                                    # print OUT $count."\t".$1."\t";
                                    debug("request $1\n");
                                    push(@{$DATA{'req_rate'}},$1);
                                    }
                        if (/^Reply rate .*min (\d+\.\d) avg
(\d+\.\d) max (\d+\.\d) stddev (\d+\.\d)/) {
                                    # print OUT $2."\t";
                                    debug("reply $2\n");
                                    push(@{$DATA{'rep_rate'}},$2);
                                    }
                         if (/^Errors: total (\d+)/) {
                                  # print OUT $1."\t";
                                  debug("error $1\n");
                                  push(@{$DATA{'error_rate'}},$1);
                                 }
                        }
                close(IN);
               }
        # print Dumper(\%DATA);
        foreach my $key (keys %DATA){
                    my $stat=Statistics::Descriptive::Full->new();
                    $stat->add_data(@{$DATA{$key}});
                    $MEAN_DATA{$key}=$stat->mean();
                    debug("the key found is $key and mean val is
$MEAN_DATA{$key} \n");
                   }

        foreach my $key (keys %MEAN_DATA){
                    # debug("the key used for mean is $key \n");
                    print OUT $MEAN_DATA{$key}."\t";
```

```
                        }
        print OUT "\n";
        $count+=$rate;
      }
close(OUT);

sub usage{
            # prints the correct use of this script
              print "Usage:\n";
              print "-h Usage \n";
              print "-v Verbose\n";
              print "-d Debug\n";

              print "./script [-d][-v][-h]\n";
        }

sub verbose{
            print "VERBOSE:".$_[0] if $VERBOSE;
        }

sub debug{
            print "DEBUG:".$_[0] if $DEBUG;
        }
```

A.3 Script to overload Web Server

This script is used to create overload on webserver.

```
<?php
 $num=range(1,50000);
shuffle($num);

for($i=0;$i<25000;$i++){
   $result[$i]=$num[$i];
}

sort($result);
  print_r($result);

?>
```