Thesis Title: "Dynamic bandwidth provisioning for datacenter"

Submitted by

Simon MISA

College of Science and Technology

School of Information and Communication Technology (SoICT)

Master of Science in ICT (Option: Operation Communication)

The Year 2018

Thesis Title: "Dynamic bandwidth provisioning for datacenter"

By
Simon MISA
Registration Number :217300731

A dissertation submitted in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN ICT (Option: Operational Communication)

In the College of Science and Technology

Supervisor: Dr. Gaurav BAJPAI
Co-Supervisor: Dr. Didacienne MUKANYILIGIRA

May 2018

DECLARATION

I declare that this Dissertation contains my own work except where specifically acknowledged

I declare that the project entitled “Dynamic bandwidth provisioning for datacenter” submitted to College of Science and Technology (CST), UR in partial fulfilment of the requirement for the award of the degree in Masters of ICT option Operational Communication is my own contribution and it has not been submitted anywhere for a similar award of the same degree or diploma at any University

I carried out this work under supervision of **Dr. Gaurav BAJPAI** and

Dr. Didacienne MUKANYILIGIRA

Simon MISA

217300731

Signed……..

Date……..

Certificate

*This is to certify that the project work entitled "*Dynamic bandwidth provisioning for datacenter*" is a record of original work done by Simon MISA with Reg no: 217300731 in partial fulfilment of the requirements for the award of Master of Science in Information and Communication Technology of College of Science and Technology, University of Rwanda during the academic year 2017-2018*

Dr. Gaurav BAJPAI **Mr. Dominique HALERIMANA**

Supervisor **Head Department of Information Technology**

ACKNOWLEDGEMENTS

I acknowledge that what I have achieved to this point was through the help of the Almighty God who give me the strength, courage and determination to carry out my responsibilities.

I thank also my supervisors **Dr. Gaurav BAJPAI** and **Dr. Didacienne MUKANYILIGIRA** for all their help, feedback, time and assistance in this project.

I am ever, especially indebted to wife, our parents and family, for their love and support throughout my lives.

MISA Simon

ABSTRACT

Today's data centers contain thousands of computers operating a variety of service and applications. The limited network bandwidth of data centers become the performance bottleneck, or simply say that this creates a congestion due to several requests. As every customer need to access service or application at high speed.

To guarantee the network performance, an efficient network bandwidth management can be leveraged, however, some research had been done so far to avoid or resolve these issues. The existing research mainly relies on a static or dynamic bandwidth allocation to virtual machines. This research paper focus on bandwidth management by allocating bandwidth to a specific service instead of allocating static or dynamic bandwidth to the entire virtual machine, where a service or application that is in the inoperative state will also be allocated bandwidth.

When a client or customer is accessing his virtual machine, after authentication this request is submitted to the server that will check the details of the service or application to be provided to the client. The server will classify the service or application according to the required bandwidth, and then provides enough bandwidth to that service.

As the application completes to be serviced the bandwidth will be released and be allocated to another service or application that need it to work properly.

This bandwidth management method will enable the efficient sharing of bandwidth and hence improves the network performance in data centers by allocating to all the services and applications to enough bandwidth.

Hence data centers will manage available limited bandwidth efficiently.

KEY WORDS: Data center, Bandwidth, Clients, Users, Condor, Condor-G, service, applications, data, allocation,

LIST OF SYMBOLS AND AND ABBREVIATIONS

UR: University of Rwanda

CST: College of Science and Technology

MOC: Masters of Operation and Communication

ICT: Information Communication and Technology

WAN: Wide Area Network

LAN: Local area network

DC: Data center

VM: Virtual Machine

TAG: Tenant Application Graph

IP: Internet Protocol

TCP: Transmission Communication Protocol

UDP: User Datagram Protocol

FTP: File Transfer Protocol

HTTP: Hypertext Protocol

FDDI: Fiber Distributed Data Interface

UTP: Unshielded Twisted-Pair

STP: Shielded Twisted-Pair

Cat: Category

PoE: Power over Ethernet

USB: Universal Serial Bus

IM: Instant Messaging

RAID: Redundant Array of Independent Disks

OS: Operating System

CPU: Computer Processing Unity

SaaS: Software as a Service

SOA: Service-oriented architecture

GSI: Grid Security Infrastructure

Condor-G: Condor Grid

BRP: Borrow-Return-Preempt

LIST OF SYMBOLS

A: interarrival distribution time

B: the service time distribution

C: represent the number of parallel servers

N: the system capacity

K: size of users

M: exponential type

es: No. of parallel servers

L: long-run time average of No. customers in system

λ: the arrival rate of jobs at system

ω: long-run average time spent in the system per customer

M/M/c/∞/∞: the Multiserver Queue

Pn: at steady state probability of having n customers in the system

μ: the service rate of one server

ρ: the server utilization

Sn: service time of nth customer

$\mathbf{L_Q\ (t)}$: the number of customers in queue at time t

W: the Long run average time spent in system by customer

$\mathbf{W_Q}$: Long-run average time spent in queue by customers

P: the server utilization

μ: the service rate of one server

TABLES OF FIGURES

LIST OF TABLES

Contents

CHAPTER ONE: INTRODUCTION

Introduction

Existing methods on bandwidth allocation can be generally classified into two folds: static bandwidth allocation and dynamic bandwidth allocation. However, none of them can address the problem of managing the network bandwidth in a data center with a mixed variety of applications [1].

Regarding the static bandwidth allocation, some methods focus on reserving bandwidth during the virtual machine (VM) placement, while some methods focus on the bandwidth allocation after VM placement [1]

There are different existing proposals for sharing cloud networks, [4, 5–7,8, 9, 10] which do not achieve many goals simultaneously
On one hand SecondNet [16] proposal is based on reserving bandwidth for each VM-to-VM part and on the other hand Lee et al. propose its new model TAG (tenant application graph) with reservation of bandwidth at the application level [17].

Faircloud shows other three methods based to allocate bandwidth on congested links, which can achieve the VM-pair fairness [11].
Guo et al. presents an allocation strategy by using on game theory and this theory which keeps fairness among different VMs [26].
NetShare [12] provides tenant level fairness on congested links and achieves proportional bandwidth sharing by using weighted fair queues. Chen et al. focus on application-level fairness and they introduce a rigorous definition of performance-centric fairness with the guiding principle that the performance of data parallel applications should be proportional to their weights [19]

Popa et al. present Elastic-Switch to dynamic utilize the spare bandwidth to the newly coming flows [13]. It can be fully implemented in hypervisors, but it has fluctuations under burst traffic when the rate limit is beyond the guarantee. Guo et al. take advantage of the Logistic Model to

design a novel distributed bandwidth allocation algorithm, with the aim of coping with highly dynamic traffic in the data center network [14].
To perform bandwidth allocation, most of them rely on a technique of rate limit on the endpoints. However, such rate limit has significant limitations: since each network flow traverses through multiple switches along its routing path, such rate limit is unaware of the in-network status of data centers. This eventually hurts the performance of applications.

In this research, another way of allocated bandwidth will show how this can be addressed differently by allocating bandwidth to the only specific services which really need it to work. Our application will not provide bandwidth to services or application which are not ready to start working directly. Service or applications which finish with bandwidth or in idle state will return it to the system so that It can be used by other application, which means no application or service will just hold bandwidth while it is not in use.

Motivation

Information technology is improving very fast and needs accessibility and speed to move at the frequency. People who keep their data on cloud need the unlimited capacity to access their data. Research has shown that many applications/services are allocated static or dynamic bandwidth either are operational or not. In this project only operational services or the ones need bandwidth to work will be provided bandwidth.

Background of the research project

Data centers host thousands of computers serving different application and services to thousands of users. The limited bandwidth of data centers become the performance bottleneck as users need to access service or application at high speed.

Some research had been done so far to avoid or resolve these issues. Previous works on bandwidth allocation mainly relies on static allocation or dynamic allocation, yet has significant limitations in managing the in network bandwidth in data centers. On the one hand, the static bandwidth allocation fairly allocates bandwidth to different entities. Such entities can be VM,

VM-pair, or tenant. Once the bandwidth is allocated, each entity will hold the bandwidth until its data transferring is completed.

Using Borrow-Return-Preempt (BRP) bandwidth management method to allocate bandwidth to the individual bandwidth requests of each application type With the BRP method, the unused fraction of minimal reserved bandwidth of an application type can be borrowed by bandwidth requests from other types of applications. The bandwidth of an application type borrowed by other application types can be returned to this application type

Problem statement

In general, Internet rate is the most important thing to be considered when running a cloud computing/Datacenter. Everyone would like to access their servers or applications at a high speed. This new model will use the GRID to allocate shared bandwidth to operating service or application efficiently to improve network performance. The data center will no longer suffer from network congestion or bottleneck when clients accessing the data within the data center.

Objectives

2.5.1 General Objective

As a postgraduate student pursuing a Masters of Operation and Communication (MOC) degree in ICT, we are expected to do a research project in order to accomplish our program and help the community on different issues using ICT, in our case the issue of bandwidth management in the data center.

2.5.2 Specific Objectives

- ✓ Use Condon G and GRID applications for bandwidth management
- ✓ Allocate bandwidth to only operational services or applications
- ✓ Clarify how this system is better than the existing one

Limitation of the research project

This research project will be limited to the allocation of limited and small bandwidth to specific services or applications using the GRID in a simulated environment. This will allow to demonstration of this technique using virtual environment instead of physical device.

Organization of the study

This project is divided into five chapters;

Chapter one is an introductory chapter, showing the introduction of the project, problem statement, Objectives, Limitation of the research project, Justification, organization of the study.

Chapter two is the literature review which describes what others have done related to the current project. It also justifies the use of solution techniques and problem-solving procedures in the project work.

Chapter three consists of research methods and procedures. It also presents the project design and how it was executed.

Chapter four presents results of this project in relation to the project goal

Chapter five is the last one that presents conclusions and recommendations based on the research experience.

Hardware and Software requirements

1. Server Machine
2. Client machines (3)
3. Condon G
4. GRID
5. Youtube downloader to client machines

Conclusion of chapter one

In this chapter, the brief description of the project has stated including the objectives, problem statement and the techniques or methodologies that will be used to run a simulation for getting the results. It also states the main assumptions that the project are planning to achieve at the end of the design and implementation. The related works and literature review are detailed the following chapter

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

This chapter of literature review provide in details the work done by other researchers, and then elaborate an overview of definitions, characteristics of concepts and technologies for dynamic bandwidth allocation in Data centre

2.2 Bandwidth

According to Shannon–Hartley theorem, bandwidth is the maximum rate at which information can be transmitted over a communications channel in the presence of noise. It is an application of the noisy-channel coding theorem to the archetypal case of a continuous-time analog communications channel subject to Gaussian noise.

The theorem establishes Shannon's channel capacity for such a communication link, a bound on the maximum amount of error-free information per time unit that can be transmitted with a specified bandwidth in the presence of the noise interference, assuming that the signal power is bounded, and that the Gaussian noise process is characterized by a known power or power spectral density. The law is named after Claude Shannon and Ralph Hartley. [18]

To set up a data center the first item to consider is bandwidth as data, virtual machines, applications and services hosted in data centre need to be accessible at high transmission rate.

Allocating bandwidth means how much portion of Internet speed you may set for a particular application, service or virtual machine to work properly

Bandwidth allocation can be static or dynamic. The static bandwidth allocation is done manually and specify a particular amount of bandwidth set for particular application. In dynamic bandwidth allocation, traffic bandwidth in a shared telecommunication medium and can be allocated on demand with fairly between different users of that bandwidth. Dynamic bandwidth allocation takes advantage of several attributes of shared networks:

(1) All users are typically not connected to the network at one time

(2) Even when connected, users are not transmitting data (or voice or video) at all times

(3) Most traffic occurs in bursts as there are gaps between packets of information that can be filled with other user traffic

2.2.2 Physical mediums

In data communications, physical medium is the transmission path over which a signal propagates. Many transmissions media are used as channels to allow communication of connected computers or other devices on a network. Each transmission medium requires specialized network hardware that has to be compatible with that medium. For telecommunications purposes, transmission media are classified as following:

- Guided: where transmission are guided along a solid medium such as a transmission line.
- Wireless: where transmission and reception are achieved by means of an antenna.

In general, networks use combinations of media which are categorized in three main types:

- **Copper cable:** Types of cable include unshielded twisted-pair (UTP), shielded twisted-pair (STP), and coaxial cable. Copper-based cables are inexpensive and easy to work with compared to fiber-optic cables,
- **Wireless:** Wireless media include radio frequencies, microwave, satellite, and infrared. Deployment of wireless media is faster and less costly than deployment of cable, particularly where there is little or no existing infrastructure
- **Fiber optics:** Fiber offers enormous bandwidth, immunity to many types of interference and noise, and improved security. Therefore, fiber provides very clear communications and a relatively noise-free environment. The downside of fiber is that it is costly to purchase and deploy because it requires specialized equipment and techniques. [10]

2.2.1 Types of cables:

According to Bradley Mitchell, many computer networks in the 21st century still rely on physical medium cables for devices to transfer data. There are several types of network standard and each designed for specific purposes.

2.2.1.1 Coaxial Cables

Coaxial cable a standard for 10 Mbps Ethernet cables, invented in 1880s used to connect televisions. It utilized one of two kinds of coax cable - thinnet (10BASE2 standard) or thicknet (10BASE5).

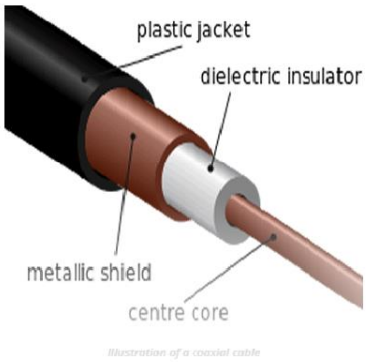


Figure 2 1 Coaxial cable

2.1.3.2 Twisted Pair Cables

Two primary types of twisted pair cable industry standards have been defined: Unshielded Twisted Pair (UTP) and Shielded Twisted Pair (STP)**.** Modern Ethernet cables use UTP wiring and it is not expensive. STP cabling are used in different types of networks like Fiber Distributed Data Interface (FDDI).

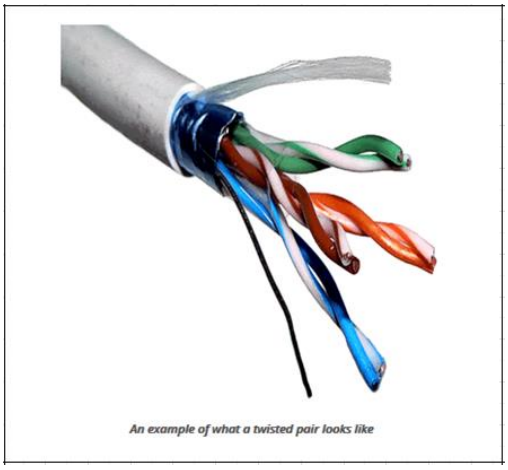


Figure 2 2 Twisted pair

Twisted pair is a standard cable for Ethernet, starting with 10 Mbps, later followed by improved versions for 100 Mbps, Cat5, and Cat5e then higher speeds up to 10 Gbps

Category 5 (Cat5)

Cat5 cabling was standard in 1995. If you're utilizing Cat5—and doing fine for your IT needs—don't fix what's not broken. If you plan to expand into more advanced IT technologies, requirements may dictate your choice.

Category 5e (Cat5e)

It has copper cable that uses a new standard; 4-twisted pairs, with all eight (8) contacts. Cat5e reduces noise and signal interference, increasing rated transfer speeds to 350 Mbit/s over 100 meters. An optimized encoding scheme allows up to 50-meter lengths of Cat5e cable to perform at Gigabit Ethernet (1000BASE-T) speeds.

Category 6 (Cat6)

The Cat6 has been around since 2002. It's predicted that upgrading to Cat6a Ethernet cable will be necessary for quite some time.

Cat6 cable is highly recommended for Power over Ethernet (PoE) and Audio/Video (AV) applications. The tighter specifications guarantee that 100-meter runs of Category 6(1000BASE-T) are capable of 1000 Mbit/s transfer speeds.

Category 6a (Cat6a)

Cat6a cabling has augmented specification designed to double transmission frequency to 500 MHz. This cable infrastructure supports full 10-Gigabit Ethernet speeds without giving up 100 meters of cable length. The same is true for Shielded Twisted Pair (STP) cables that reduce alien crosstalk.

Most computers are linked at Gigabit Ethernet speeds. IEEE 802.3 (10GBASE-T standard), continues to drive demand for high performance. Reaching 10,000 Mb/s requires a higher category of cable, such as Cat6 or Cat6a.

Category 7 (Cat7)

Designed for Gigabit Ethernet. While Cat7 may offer more than needed, each newer cable standard allows user higher speeds with lower crosstalk, even with longer cables. Most businesses still have no need for updating their hardware to Cat7 Ethernet cable, much less Cat7a or Cat8 cables,

Ethernet twisted pair cables contain up to eight (8) wires wound together in pairs to minimize electromagnetic interference [21].

2.1.4 Fiber Optics

Instead of insulated metal wires transmitting electrical signals, fiber optic network cables work using strands of glass and pulses of light. These network cables are bendable despite being made of glass. They have proven especially useful in wide area network (WAN) installations where long distance underground or outdoor cable runs are required and also in office buildings where a high volume of communication traffic is common.

Two Types of Fibre Optic Cables:

Single mode: Deliver 10 Gigabit Ethernet at 40,000 meters. Expensive and difficult to work with, this cable is so narrow that light can travel through it only in a single path.

Multimode: Delivers 10 Gigabit Ethernet at 550 meters. Wider core diameters give light beams the freedom to travel in multi-paths, causing signal distortion at its receiving end [27].

2.1.5 USB Cables

Most Universal Serial Bus (USB) cables connect a computer with a peripheral device (keyboard or mouse) rather than to another computer. However, special network adapters (sometimes called dongles) also allow connecting an Ethernet cable to a USB port indirectly. USB cables feature twisted pair wiring.

2.1.6 Serial and Parallel Cables

Many PCs in the 1980s and early 1990s lacked Ethernet capability, and USB had not been developed yet, serial and parallel interfaces (now obsolete on modern computers) were sometimes used for PC-to-PC networking. So-called null model cables, for example, connected the serial ports of two PCs enabling data transfers at speeds between 0.115 and 0.45 Mbps.

2.1.7 Crossover Cables

A crossover cable joins two network devices of the same type, such as two PCs or two network switches. The use of Ethernet crossover cables was especially common on older home networks years ago when connecting two PCs directly together. Externally, Ethernet crossover cables appear nearly identical to ordinary the only visible difference being the order of color-coded wires appearing on the cable's end connector. Manufacturers typically applied special distinguishing marks to their crossover cables for this reason. [20]

2.3 Servers

A Server is a combination of hardware or software designed to provide services to clients. When used alone, the term typically refers to a computer which may be running a server operating system, but is commonly used to refer to any software or dedicated hardware capable of providing services. Servers are often dedicated, meaning that they perform no other tasks besides their server tasks. On multiprocessing operating systems, however, a single computer can execute several programs at once. A server in this case could refer to the program that is managing resources rather than the entire computer.

Network server is a computer that manages network traffic, it is designed to process requests and deliver data to other computers (Clients) over a local network or the Internet.

Network servers typically are configured with additional processing, memory and storage capacity to handle the load of servicing clients

Server types

This list categorizes the many different types of servers used in the marketplace today.

2.3.1 Proxy Server

A proxy server sits between a client program (typically a Web browser) and an external server (typically another server on the Web) to filter requests, improve performance, and share connections.

2.3.2 Mail Server

Almost as ubiquitous and crucial as Web servers, mail servers move and store mail over corporate networks (via LANs and WANs) and across the Internet.

2.3.3 Server Platforms

A term often used synonymously with operating system, a platform is the underlying hardware or software for a system and is thus the engine that drives the server.

2.3.4 Web Server

At its core, a Web server serves static content to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. This entire exchange is mediated by the browser and server talking to each other using HTTP.

2.3.5 Application Server

Sometimes referred to as a type of middleware, application servers occupy a large chunk of computing territory between database servers and the end user, and they often connect the two.

2.3.6 Real-Time Communication Server

Real-time communication servers, formerly known as chat servers or IRC Servers, and still sometimes referred to as instant messaging (IM) servers, enable large numbers users to exchange information near instantaneously.

2.3.7 FTP Server

One of the oldest of the Internet services, File Transfer Protocol makes it possible to move one or more files securely between computers while providing file security and organization as well as transfer control.

2.3.8 Collaboration Server

In many ways, collaboration software, once called 'groupware,' demonstrates the original power of the Web. Collaboration software designed to enable users to collaborate, regardless of location, via the Internet or a corporate intranet and to work together in a virtual atmosphere.

2.3.9 List Server

List servers offer a way to better manage mailing lists, whether they be interactive discussions open to the public or one-way lists that deliver announcements, newsletters or advertising.

2.3.10 Telnet Server

A Telnet server enables users to log on to a host computer and perform tasks as if they're working on the remote computer itself.

2.3.11 Open Source Server

From your underlying open source server operating system to the server software that help you get your job done, open source software is a critical part of many IT infrastructures.

2.3.12 Virtual Server

In 2009, the number of virtual servers deployed exceeded the number of physical servers. Today, server virtualization has become near ubiquitous in the data center [23].

2.4 Storages

In August 2010, Ramesh Natarajan has explained the Diagram of RAID 0, RAID 1, RAID 5 and RAID 10.

RAID stands for Redundant Array of Inexpensive (Independent) Disks.On most situations you will be using one of the following four levels of RAIDs.

- RAID 0
- RAID 1
- RAID 5
- RAID 10 (also known as RAID 1+0)

This article explains the main difference between these raid levels along with an easy to understand diagram.
In all the diagrams mentioned below:

- A, B, C, D, E and F – represents blocks
- p1, p2, and p3 – represents parity

2.4.1 RAID LEVEL 0

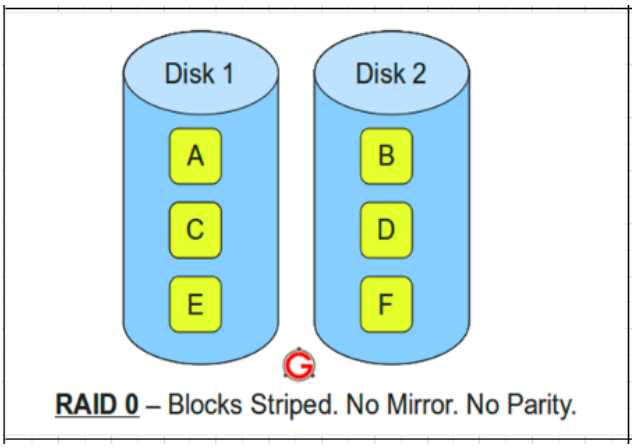


Figure 2 3 RAID Level 0

Following are the key points to remember for RAID level 0.

- Minimum 2 disks.

- Excellent performance (as blocks are striped).
- No redundancy (no mirror, no parity).
- Don’t use this for any critical system.

2.4.2 RAID LEVEL 1

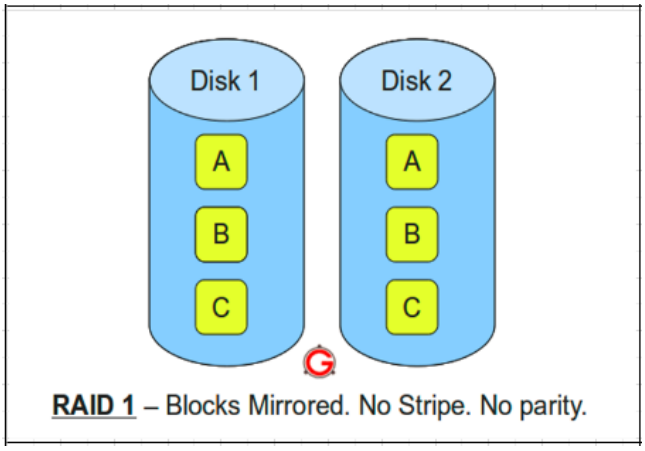


Figure 2 4 RAID Level 1

Following are the key points to remember for RAID level 1.

- Minimum 2 disks.
- Good performance (no striping. no parity).
- Excellent redundancy as blocks are mirrored).

2.4.3 RAID LEVEL 5

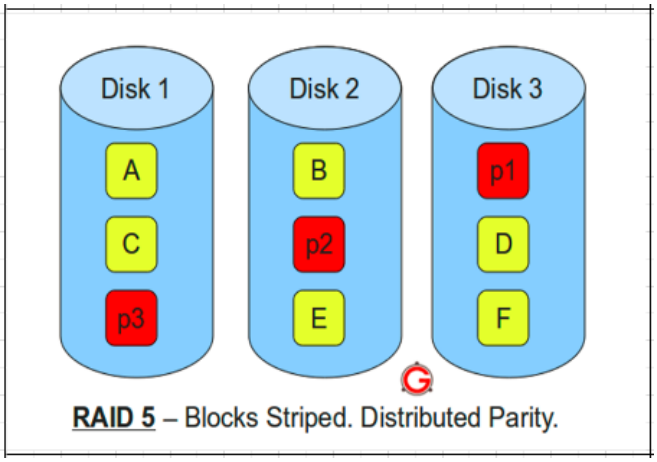


Figure 2 5 RAID Level 5

Following are the key points to remember for RAID level 5.

- Minimum 3 disks.
- Good performance (as blocks are striped).
- Good redundancy (distributed parity).
- Best cost effective option providing both performance and redundancy. Use this for DB that is heavily read oriented. Write operations will be slow.

2.4.4 RAID LEVEL 10

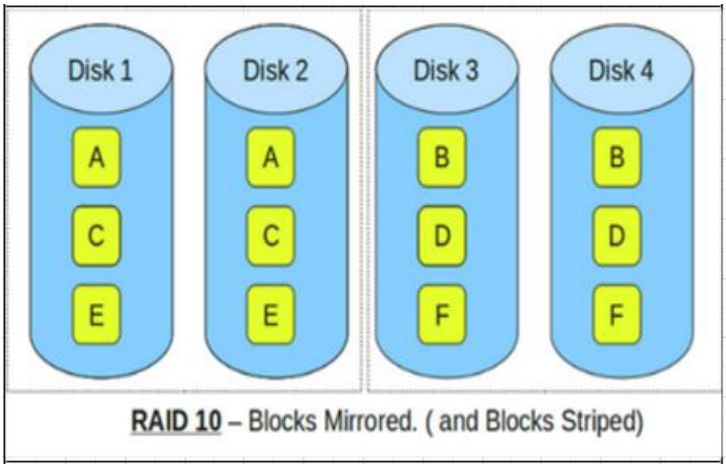


Figure 2 6 RAID 10 Level

Following are the key points to remember for RAID level 10.

- Minimum 4 disks.
- This is also called as “stripe of mirrors”
- Excellent redundancy (as blocks are mirrored)
- Excellent performance (as blocks are striped)
- If you can afford the dollar, this is the BEST option for any mission critical applications (especially databases).

2.5 Operating System (OS)

An operating system (commonly abbreviated to either *OS* or *O/S*) is an interface between hardware and user; an OS is responsible for the management and coordination of activities and the sharing of the resources of the computer. The operating system acts as a host for computing applications that are run on the machine.

As a host, one of the purposes of an operating system is to handle the details of the operation of the hardware.

There are some popular server operating systems such as Free BSD, Solaris, Windows and Linux

2.6 Hardware

2.6.1 Router

A device to interconnect heterogeneous network technologies, using the most general information possible. Routers examine packet information that is link independent (to support heterogeneous networks), but also data independent (to support heterogeneous applications). Contrast with *bridge* and *gateway*

2.6.2 Switches

2.6.2.1 Core switches

The Core switches perform routing at Layer 3 (the network layer) and switching at Layer 2 (the data link layer that moves data across the physical links of a network).

Core switches are high-throughput, high-performance packet and frame movers. Packets and frames are simply moved from one core switch to another core switch, and eventually down to the next tier of switches which is the distribution tier.

2.6.2.2 Distribution switches

The distribution tier addresses a new set of unique switching needs, and is the workhorse of any enterprise network.

Distribution switches are used to connect the core and access tiers together on the network. If data needs to be moved from one distribution block to another, the switch pushes that data up to the core switches, which know the optimal path to the destination distribution tier switch.

Distribution switches also interconnect all network access tier switches. Because there are so many interconnections in a network, distribution switches have higher port density than core switches, which have far fewer interconnections to other switches.

Distribution switches also enforce all forms of network policies. Access lists are configured and implemented in the distribution tier to permit or deny traffic from one network to another. Quality of service policies are also found here to prioritize packets and put them into pre-defined queues for optimal transport of time-sensitive information. In addition to port density, distribution tier switches must have enough CPU speed and memory to perform all tasks at or near wire speed.

2.6.2.3 Access switches

At the bottom of the classic three-tier switch design is the access tier. Access tier switches are the only ones that directly interact with end-user devices. Because an access switch connects the majority of devices to the network, the access tier typically has the highest port density of all switch types.

Despite the high port-count, however, access switches usually provide the lowest throughput-per-port of all switches. For example, most modern access switches provide a 10/100/1000 Mbps copper Ethernet connection to end devices. By contrast, core and distribution tier switches commonly use between 10 Gbps and 100 Gbps fiber-optic ports.

So in terms of CPU and raw throughput, access switches are on the low end of the scale. But these switches offer many features that cater specifically to end-devices that the upper tiers do not

require. For example, access switches commonly support Power over Ethernet, which can power many endpoint devices, including wireless access points and security cameras.

Additionally, access switches are better able to interact with endpoints from a security perspective. Things like port-security, 802.1X authentication and other security mechanisms are built directly onto access switch software. [17]

2.2.1 Data center

A data center (or datacenter) is a room that houses computing facilities like servers, routers, switches and firewalls, as well as supporting components like backup equipment, fire suppression facilities and air conditioning that different businesses or organizations use to organize, process, store and disseminate large amounts of data.

A business typically relies heavily upon the applications, services and data contained within a data center, making it a focal point and critical asset for everyday operations. It can be private or shared.

When data centers are shared, virtual data center access often makes more sense than granting total physical access to various organizations and personnel. Shared data centers are usually owned and maintained by one organization that leases center partitions (virtual or physical) to other client organizations. Often, client/leasing organizations are small companies without the financial and technical resources required for dedicated data center maintenance. The leasing option allows smaller organizations to obtain professional data center advantages without heavy capital expenditure.

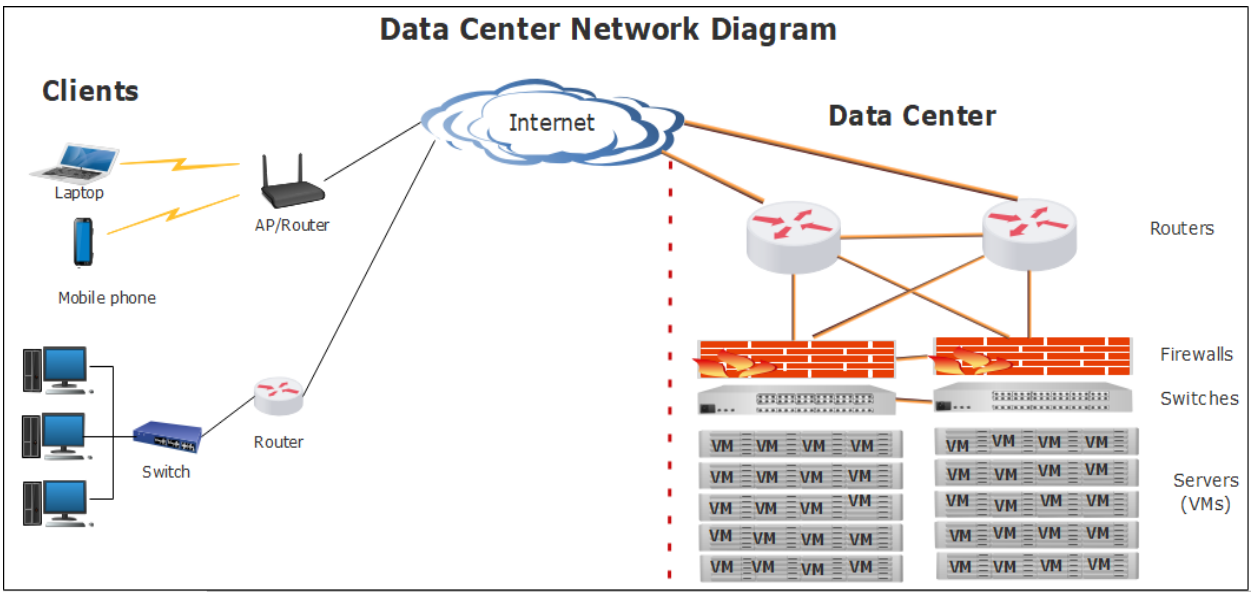


Figure 2 7 Data center architecture

Architectural Design of Computing Clouds

This section presents basic cloud design principles. We start with a basic cloud architecture to process massive data with a high-degree parallelism. Then we study virtualization support, resource provisioning, infrastructure management, and then performance modelling.

2.2.2 Cloud Architecture for Distributed Computing

An Internet cloud is envisioned as a public cluster of servers provisioned on demand to perform collective web services or distributed applications using the data center resources. The cloud design objectives are first specified below. Then we present a basic cloud architecture design:

Cloud Platform Design Goals: *Scalability, virtualization, efficiency,* and *reliability* are four major design goals of a cloud computing platform. Clouds support Web 2.0 applications. The cloud management receives the user request and then finds the correct resources, and then calls the provisioning services which invoke resources in the cloud. The cloud management software needs to support both physical and virtual machines [3]. Security in shared resources and shared access of datacenters also post another design challenge. The platform needs to establish a very large-scale HPC infrastructure. The hardware and software systems are combined together to make it

easy and efficient to operate. The system scalability can benefit from cluster architecture. If one service takes a lot of processing power or storage capacity or network traffic, it is simple to add more servers and bandwidth. The system reliability can benefit from this architecture. Data can be put into multiple locations. For example, the user email can be put in three disks which expand to different geographical separate data centers. In such situation, even one of the datacenters crashes, the user data is still accessible. The scale of cloud architecture can be easily expanded by adding more servers and enlarging the network connectivity accordingly.

Enabling Technologies for Clouds: The key driving forces behind cloud computing are the ubiquity of broadband and wireless networking, falling storage costs, and progressive improvements in Internet computing software. Cloud users are able to demand more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity, whereas service providers can increase the system utilization via multiplexing, virtualization, and dynamic resource provisioning. Clouds are enabled by the progress in hardware, software and networking technologies summarized in Table 2.1.

Table 2 1 Cloud Enabling Technologies in Hardware, Software, and Networking

Technology Requirements and Benefits	**Technology Requirements and Benefits**
Fast Platform	Fast Platform
Deployment	Deployment
Fast, efficient, and flexible deployment of cloud resources to provide	Fast, efficient, and flexible deployment of cloud resources to provide
dynamic computing environment to users	dynamic computing environment to users
Virtual Clusters	Virtual Clusters
on Demand	on Demand
Virtualized cluster of VMs provisioned to satisfy user demand and virtual	Virtualized cluster of VMs provisioned to satisfy user demand and virtual

These technologies play instrumental roles to make cloud computing a reality. Most of these technologies are mature today to meet the increasing demand. In the hardware area, the rapid progress in multi-core CPUs, memory chips, and disk arrays has made it possible to build faster data centers with huge storage space. Resource virtualization enables rapid cloud deployment faster and fast disaster recovery. *Service-oriented architecture* (SOA) also plays a vital role. The progress in providing *Software as a Service* (SaaS), Wed.2.0 standards, and Internet performance have all contributed to the emergence of cloud services. Today's clouds are designed to serve a large number of tenants over massive volume of data. The availability of large-scale, distributed storage systems lies the foundation of today's data centers. Of course, cloud computing is greatly benefitted by the progress made in license management and automatic billing techniques in recent years

A Generic Cloud Architecture: A security-aware cloud architecture. The Internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using datacenter resources. Cloud platform is formed dynamically by provisioning or de-provisioning, of servers, software, and database resources. Servers in the cloud can be physical machines or virtual machines. User interfaces are applied to request services. The provisioning tool carves out the systems from the cloud to deliver on the requested service.

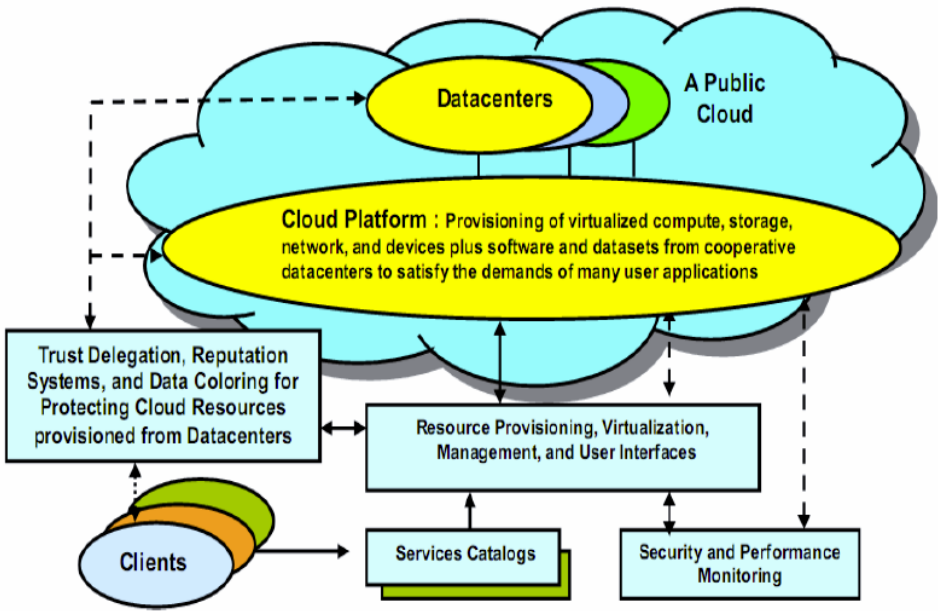


Figure 2 8 Security-aware cloud platform, virtual cluster of VMs, storage, and networking resources over DC servers operated by providers.

In addition to building the server cluster, cloud platform demand distributed storage and accompanying services. The cloud computing resources are built in datacenters, which are typically owned and operated by a third-party provider. Consumers do not need to know the underlying technologies. In a cloud, software becomes a service. The cloud demands a high-degree of trust of massive data retrieved from large datacenters. We need to build a framework to process large scale data stored in the storage system.

This demands a distributed file system over the database system. Other cloud resources are added into a cloud platform including the storage area networks, database systems, firewalls and security devices. Web service providers offer special APIs that enable developers to exploit Internet clouds. Monitoring and metering units are used to track the usage and performance of resources provisioned

The software infrastructure of a cloud platform must handle all resource management and do most of the maintenance, automatically. Software must detect the status of each node, server joining and leaving and do the tasks accordingly. Cloud computing providers, like Google and Microsoft, have built a large number of datacenters all over the world. Each datacenter may have thousands of servers The location of the datacenter is chosen to reduce power and cooling costs. Thus, the datacenters are often built around hydroelectricity power supply. The cloud physical platform builder concerns more about the performance/price ratio and reliability issues than the shear speed performance.

In general, the private clouds are easier to manage. Public clouds are easier to access. The trends of cloud development are that more and more clouds will be hybrid. This is due to the fact that many cloud applications must go beyond the boundary of an Intranet. One must learn how to create a private cloud and how to interact with the public clouds in the open Internet. Security becomes a critical issue in safeguard the operations of all cloud types. We will study the security and privacy issues of cloud services [24].

2.2.3 Topology

Typical architectures today consist of either two- or three-level trees of switches or routers. A three-tiered design (see Figure 1) has a *core* tier in the root of the tree, an *aggregation* tier in the middle and an *edge* tier at the leaves of the tree. A two-tiered design has only the core and the edge tiers. Typically, a two-tiered design can support between 5K to 8K hosts. Since, the target is approximately 25,000 hosts, restriction to the three-tier design is followed.

Switches1 at the leaves of the tree have some number of GigE ports (48–288) as well as some number of 10 GigE uplinks to one or more layers of network elements that aggregate and transfer packets between the leaf switches. In the higher levels of the hierarchy there are switches with 10 GigE ports (typically 32–128) and significant switching capacity to aggregate traffic between the edges.

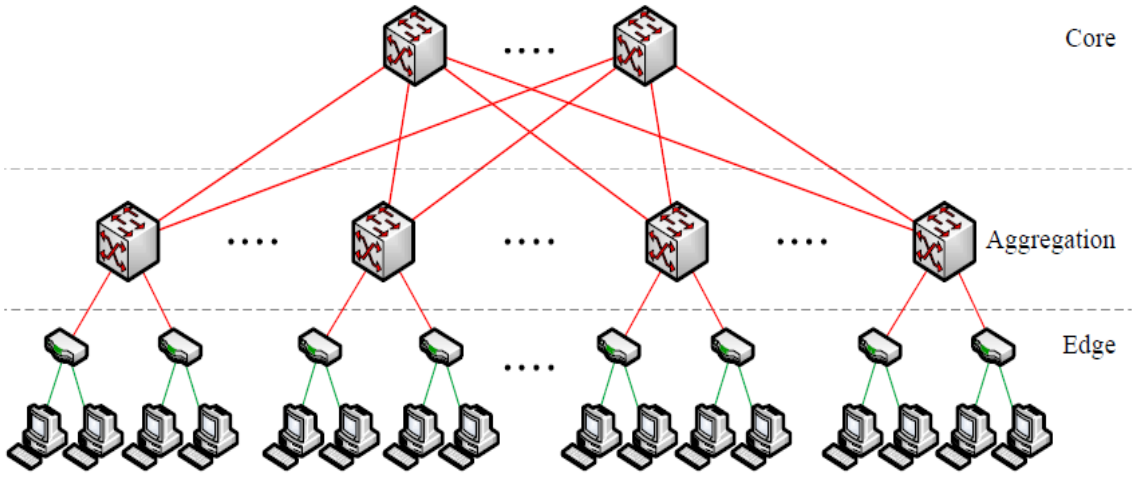


Figure 2 9 DC 3Tier topology. With Host to switch links GigE and switches are 10 GigE

Two-Level Routing Table

To provide the even-distribution mechanism motivated. We modify routing tables to allow two-level prefix lookup.

Each entry in the main routing table will potentially have an additional pointer to a small secondary table of *(suffix, port)* entries. A first-level prefix is *terminating* if it does not contain any second level suffixes, and a secondary table may be pointed to by more than one first-level prefix. Whereas entries in the primary table are left-handed (i.e., */m prefix* masks of the form 1m032−m), entries in

the secondary tables are right-handed (i.e. */m suffix* masks of the form 032−*m*1*m*). If the longest-matching prefix search yields a non-terminating prefix, then the longest-matching suffix in the secondary table is found and used.

This two-level structure will slightly increase the routing table lookup latency [28], but the parallel nature of prefix search in hardware should ensure only a marginal penalty. This is helped by the fact that these tables are meant to be very small. As shown below, the routing table of any pod switch will contain no more than *k*/2 prefixes and *k*/2 suffixes.

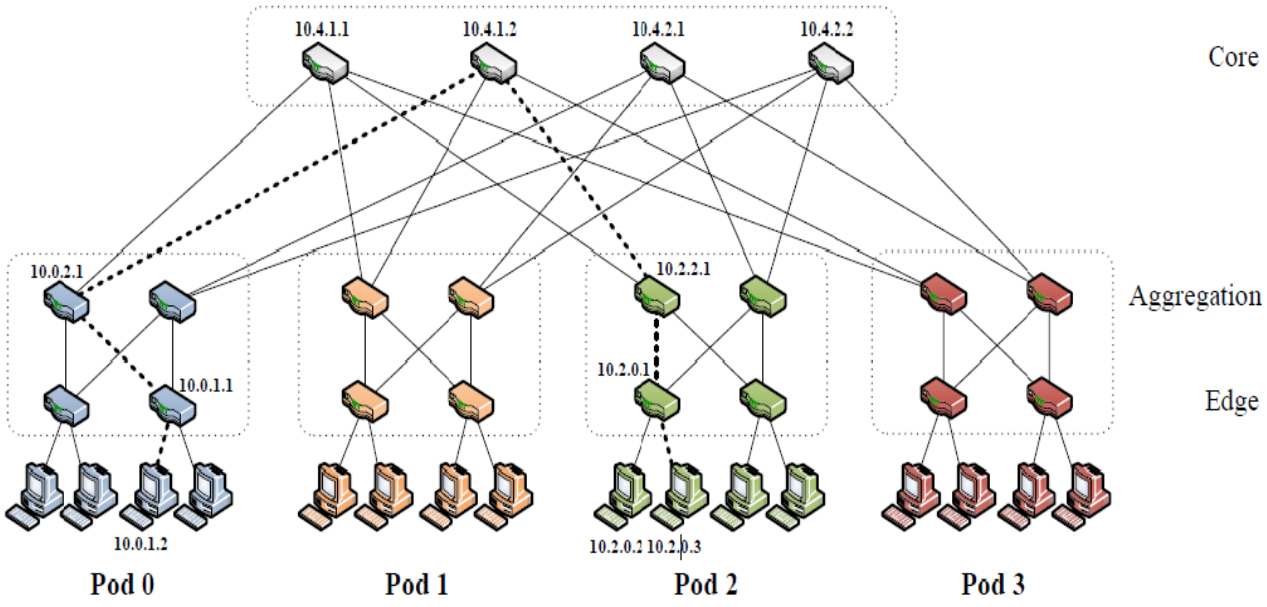


Figure 2 10 DC 2Tier topology and packets from 10.0.1.2 to 10.2.0.3 would take the dashed path.

2.3 Grid computing

Grid computing is a processor architecture that combines computer resources from various domains to reach a main objective [34] In grid computing, the computers on the network can work on a task together, thus functioning as a supercomputer [2][30]

Typically, a grid works on various tasks within a network, but it is also capable of working on specialized applications. It is designed to solve problems that are too big for a supercomputer while

maintaining the flexibility to process numerous smaller problems. Computing grids deliver a multiuser infrastructure that accommodates the discontinuous demands of large information processing.

Grid computing is a collection of computers working together to perform various tasks. It distributes the workload across multiple systems, allowing computers to contribute their individual resources to a common goal. [31]

A computing grid is similar to a cluster [29] [32] but each system (or node) on a grid has its own resource manager. In a cluster, the resources are centrally managed, typically by a single system. Additionally, clusters are usually located in a single physical space (such as a LAN), whereas grid computing often incorporates systems in several different locations (such as a WAN).

Different Grid computing technologies have been developed by the Globus Project to provide different standard protocols and services for remote resource accessibility.
Most of those technologies include Grid Resource Allocation Manager (GRAM), Grid Security Infrastructure (GSI), and Global Access to Secondary Storage (GASS). Those protocols can be used mainly to construct secure remote execution systems which cross boundaries of institutions with the use of existing batch systems without making modification or changes.

Condor

The traditional core components Condor system:

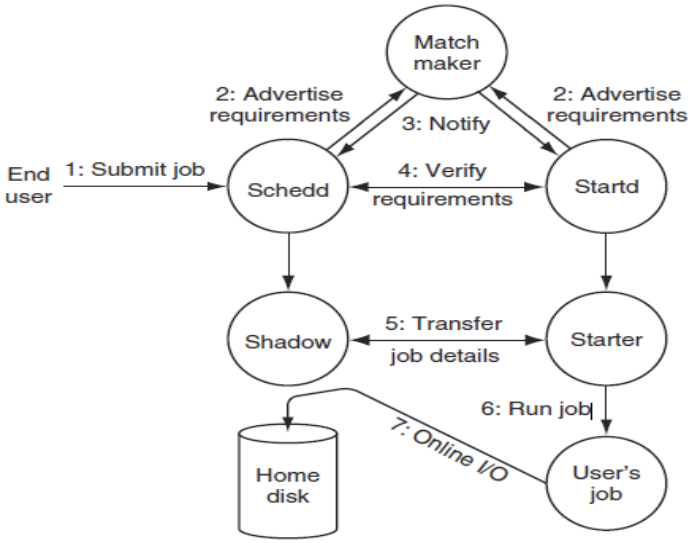


Figure 2 11 Traditional condor

This describes the responsibility of each component from the Kernel of Condor and relationship among component:

The schedd (Scheduler) is an entry point and it is in direct contact with the Grid computing services. It provides an interface for submitting, querying and removing different jobs. The schedd can also store, find the required places for them to be executed like decide where to run the job with sufficient memory and that are owned by a trusted user.

The startd is another Grid computing component responsible of finding best places, works to be done and execute them. The startd can classify into groups or categories the jobs and allow some jobs to be done like during the day and other during the nigh according to the best time. It is also monitor the state of the machine and clear everything when the job is done or no longer there.

The matchmaker introduces the compatibility of consumers which are schedds and producers which are startds. Matchmaker receives many advertisements written in ClassAd describing their current state and properties of the requested resources. It enforces system wide policies which are not enforced by schedd and startd.
These schedds and startds are collaborating to ensure that all the requirements are valid before job execution, begin working together and stop once all the required matches are no longer satisfied..

There are two subprocesses (starter and Shadow) which work together to execute the job. The starter creates an execution environment where job is done like working directory, standard I/O streams, monitor the status, inform other component of Condor about the status and so on but It does not provide policies and it relies on the shadow to decide what to run and how to do it.

The *shadow* is in charge of creating required policy decisions needed by a job.
It names the executable, arguments, environment, standard I/O streams, and all requirements necessary for completing specific job. It checks the output for determining if the job is completed or failed

The shadow also serves as a basic data server for the job and It gives remote I/O channel that the user's job may call either directly or via a proxy in the starter.

The shadow controls how and when resources are used and categorised according to their reaction to failures. *Resilient processes* are designed to handle a wide variety of error conditions via standard techniques such as retry, logging, resetting, and notification.

Brittle processes simply abort and exit upon detecting any sort of unexpected condition. As might be expected, resilient processes can be complex to develop and debug, generally requiring the management of persistent state and logging in stable storage. Brittle processes are easier to create and maintain.

Although these processes are brittle, they may still manipulate persistent state. For example, both make use of local logs to record their progress. The shadow even manipulates the job-state database stored in the schedd to indicate the disposition of the job and the amount of resources it has consumed. If either the shadow or starter should fail unexpectedly, the job state does not change.

Thus, the mere disappearance of a job cannot be considered to be evidence of its success. Neither the shadow nor the starter is responsible for clean-up. This task is left to the resilient schedd and startd, which enforce the lease that was negotiated to use the machine. Each process monitors the state of the machine on which it runs as well as the disposition of its shadow and starter children. If any child process should exit, then the resilient processes are responsible for cleaning up by killing runaway jobs, deleting temporary disk space, and discarding unused credentials.

Unlike Condor-G, no remnant of the job is left behind after a failure. This enforced clean-up ensures the consistency of the entire system. The startd must not simply restart a job after a failure, because the schedd will not necessarily know (or desire) that the job continued there. Rather, the schedd is left free to choose the next site of the job. It might try again at the location of the last failure, it might attempt to use another known but idle machine, or it may begin matchmaking all over again. The schedd is charged with this responsibility precisely because it alone knows the state of the job.

The matchmaker is also a brittle process, although it has few error conditions to encounter. If the matchmaker process or machine should crash, running schedds and startds will not be affected,

although no further matches will be made. Once restarted, a matchmaker will quickly rebuild its state from the periodic updates of startds and schedds, and the pool will operate as before.

Every machine that runs some component of Condor is managed by a resilient *master* process. This process has no duty except to ensure that schedds, startds, and matchmakers are always running. If such a service process should terminate—whether due to a bug, administrative action, or just plain malice—the master logs the event, restarts the process, and may inform the human system administrator. Repeated failures are interspersed with an exponentially increasing delay to prevent busy-looping on unexpected conditions.

This ability to start and stop processes is also used as a remote "master switch" to enable and disable Condor simultaneously on large numbers of machines. The UNIX "init" process manages the master process and thus handles system startup, shutdown, and errors in the master itself.

This dichotomy of resilient and brittle processes has several advantages. As an engineering model, it dramatically simplifies the addition of new technologies and features to serve running jobs. The vast majority of these are placed in the brittle processes, where error conditions and even faulty code are cleanly handled by the controlled destruction of the whole process. The resilient processes are naturally more complicated, but take part only in the fundamental matchmaking interaction, which is well debugged and unaffected by the addition of most new features.

However, the brittle interaction between the shadow and the starter has one significant drawback. There must be a reliable network connection between the submission and execution sites for the entire lifetime of a job. If it is broken, the job is not lost, but a significant amount of work must be repeated. To permit temporary disconnections between the two sites requires that both sides become resilient processes with persistent state and a more complex interaction. Exactly this model is explored in Condor-G.

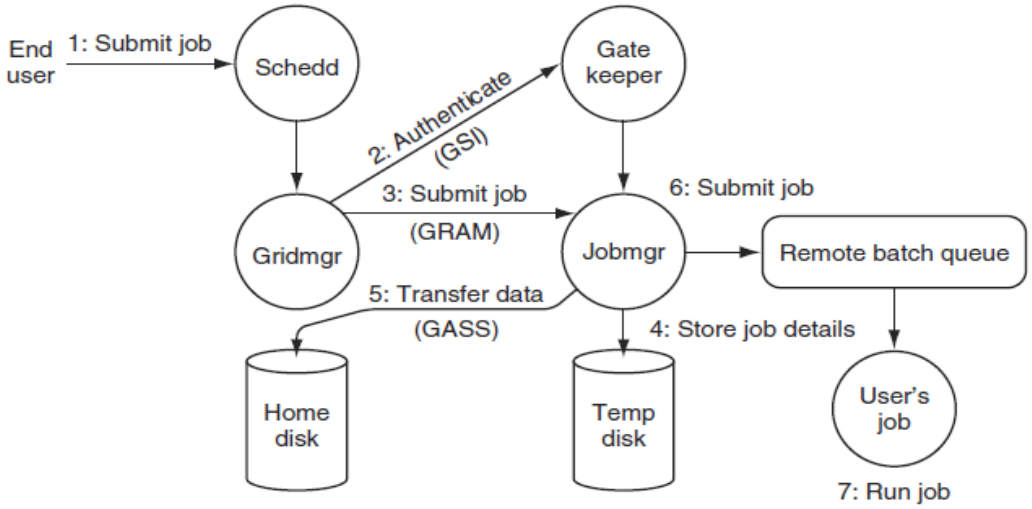


Figure 2 12 Condor-G

Condor for the Grid (Condor-G)

In the Condor-G as well as traditional Condor, the end user contact directly the *schedd*, which keeps all job state in persistent storage, specifies the general requirements each job has for an execution machine. The user must specify the name of the Globus Toolkit GRAM server that represents an entire remote batch system.

The schedd creates a process called a *gridmanager*, for each submitted job. which is responsible for contacting a remote batch queue and providing the details of the job to be run. The gridmanager is analogous to the shadow in traditional Condor. It is responsible for actively seeking out a remote batch system through its main point of contact, the *gatekeeper*. Once a job begins running, the gridmanager is also responsible for servicing the job's input and output needs through the GASS protocol.

The *gatekeeper* is responsible for enforcing the admission policies of a remote batch queue. Using GSI ( ), it accepts connections, authenticates the remote user, and maps the remote credentials into a local user ID. Once authenticated, the client may request a particular service, such as access to a batch queue through a jobmanager. The gatekeeper creates the new service process with the local user ID and passes the connection to the remote user.

The *jobmanager* is responsible for creating a remote execution environment on top of an existing batch execution system. A variety of jobmanagers are available for communicating with batch systems such as LoadLeveler (a descendant of Condor). A jobmanager could even pass jobs along to a traditional Condor pool such as that discussed in the preceding section. The jobmanager bears a responsibility similar to that of the traditional Condor starter, but it differs in one significant way: unlike the starter, it uses the local batch system to hold the details of a job in persistent storage and attempts to execute it even while disconnected from the gridmanager. This strategy permits a system to be more resilient with respect to an unreliable network, but also creates significant complications in protocol design.

The use of the GRAM protocol design to illustrate several important issues relating to remote interactions. An early version of the GRAM protocol design, used atomic interactions for the submission and completion of jobs. For example, the submission of a job from the gridmanager to the jobmanager consisted of a single message containing the job details. Upon receipt, the jobmanager would create a unique name for the job, store the details under that name, and return the ID to the gridmanager. Thereafter, it would transfer the executable and input files and attempt to submit the job to the remote batch queue.

However, a failure at any of several key points in this interaction would cause trouble. Some crashes would result in an *orphan* job, left running and consuming

CPU and storage, but with no way to stop or recover it. Other crashes would result in a *wasted* job, which ran successfully but could not communicate its results back to the submitter. [16]

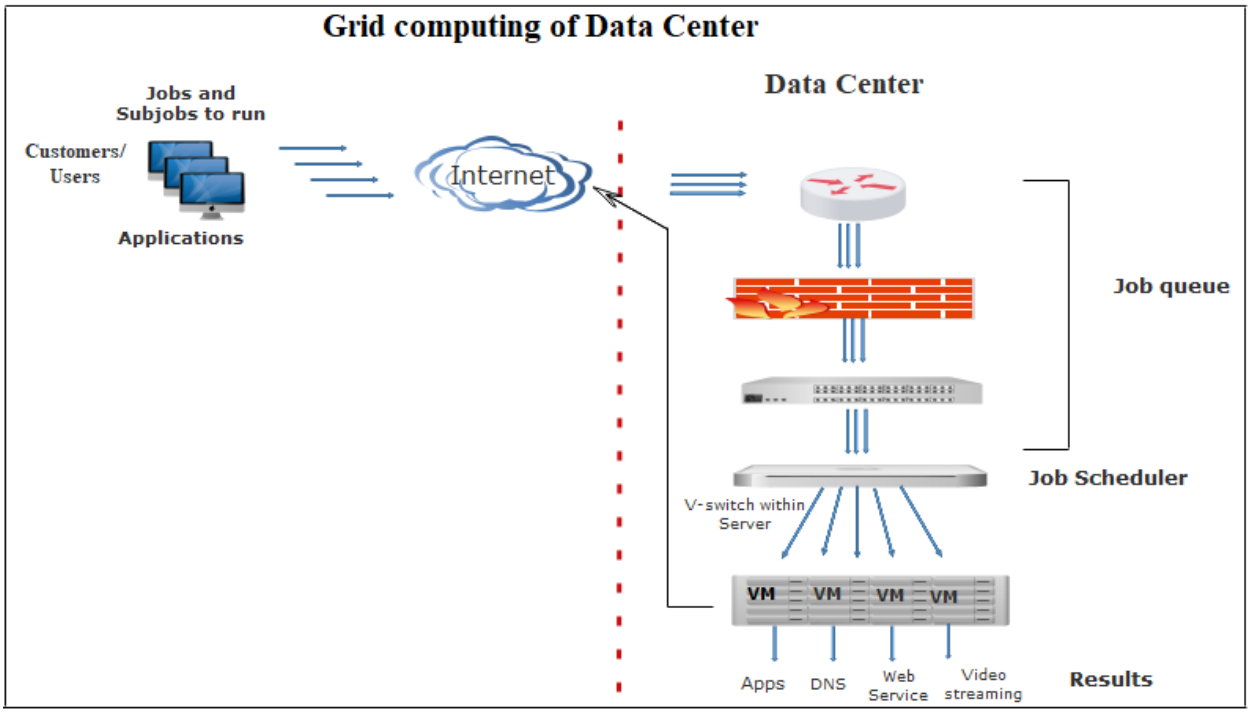


Figure 2 13 Data center jobs flow from Internet

Related Works for bandwidth allocation

Previous work on bandwidth allocation mainly relies on static allocation or dynamic allocation, yet has significant limitations in managing the in network bandwidth in data centers. On the one hand, the static bandwidth allocation fairly allocates bandwidth to different entities. Such entities can be VM [10], VM-pair [11], or tenant [12]. Once the bandwidth is allocated, each entity will hold the bandwidth until its data transferring is completed.

As a result, the intra-datacenter bandwidth cannot be efficiently utilized when entities are not able to fully utilize the bandwidth allocate to them. On the other hand, dynamic allocation is achieved by enforcing rate limit on traffic among endpoints, like VMs in virtualized datacenters [13, 14]. Unfortunately, since traffic is sent by the source, traversing through some intermediate (like, switches), and finally arrivals the destination, simply enforcing rate limit in the endpoints cannot capture the dynamic change in the intra-datacenter network.

Furthermore, it will disturb/ delay the application performance, especially the performance for some latency-sensitive applications. The reason can be that it does not provide strict priority, even low priority applications, which should be paused, continue to send packets.

To satisfy these requirements, we first present a max-min bandwidth constraint model, where the reserved bandwidth of each application type is enforced between a minimal value and a maximum value. The idle part of the minimal reserved bandwidth for each application type is allowed to be borrowed by other application types, raising the opportunity to use the in-network bandwidth in an elastic and flexible way.

Based on this model, we further present a Borrow-Return-Preempt (BRP) bandwidth management method to allocate bandwidth to the individual bandwidth requests of each application type With the BRP method, the unused fraction of minimal reserved bandwidth of an application type can be borrowed by bandwidth requests from other types of applications. The bandwidth of an application type borrowed by other application types can be returned to this application type. Moreover, a high priority bandwidth request of an application type is allowed to preempt the bandwidth from low-priority requests of both this application type and other application types. To evaluate the performance of our proposed BRP method, by conducting comprehensive simulations. The results have shown that BRP can improve the in-network bandwidth utilization by up to 19.7%, and accommodate 37% more requests with bandwidth guaranteed, compared to the conventional fair allocation method [15].

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Introduction

This chapter three describes how the research will be conducted in order to achieve the stated objectives. It will demonstrate the research design and procedures, techniques and instruments, data processing and analysis significance of the study and limitations that were encountered during the research process. The scientific methods for conducting research have been used which are both qualitative and quantitative approaches as data analysis. Different experimental research approach was also considered because several simulation results have been presented using condor G.

3.2 Grid architecture

The establishment, management, and exploitation of dynamic, cross-organizational VO sharing relationships require new technology [33]. *Grid architecture* identifies fundamental system components, specifies the purpose and function of these components, and indicates how these components interact with one another. The goal is not to provide a complete enumeration of all required components but to identify requirements for general component classes.

The result is an extensible, open architectural structure within which can be placed solutions to key VO requirements. Our architecture and the subsequent discussion organize components into layers, as shown in Figure 3.1.

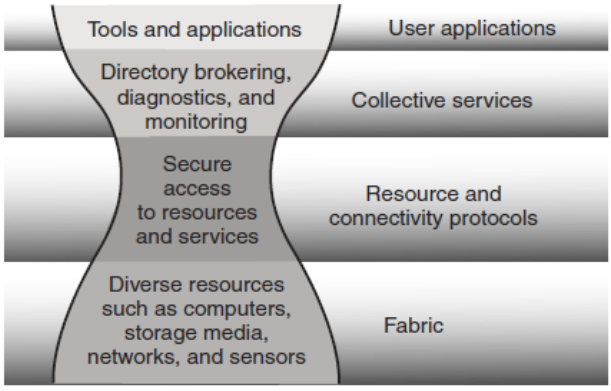


Figure 3 1 The layered Grid architecture.

Components within each layer share common characteristics but can build on capabilities and behaviours provided by any lower layer.

Our Grid architecture is based on the principles of the “hourglass model”

The narrow neck of the hourglass defines a small set of core abstractions and protocols (e.g., TCP and HTTP), onto which many different high-level behaviours can be mapped (the top of the hourglass), and which themselves can be mapped onto many different underlying technologies (the base of the hourglass). By definition, the number of protocols defined at the neck must be small. In our architecture, the neck of the hourglass consists of *resource* and *connectivity* protocols that facilitate the sharing of individual resources. Protocols at these layers are designed so that they can be implemented on top of a diverse range of resource types, defined at the *fabric* layer, and can in turn be used to construct a wide range of global services and application-specific behaviour’s at the *collective* layer—so called because they involve the coordinated (“collective”) use of multiple resources [16].

3.3 Steps of the processes

3.3.1 Sequence Diagram

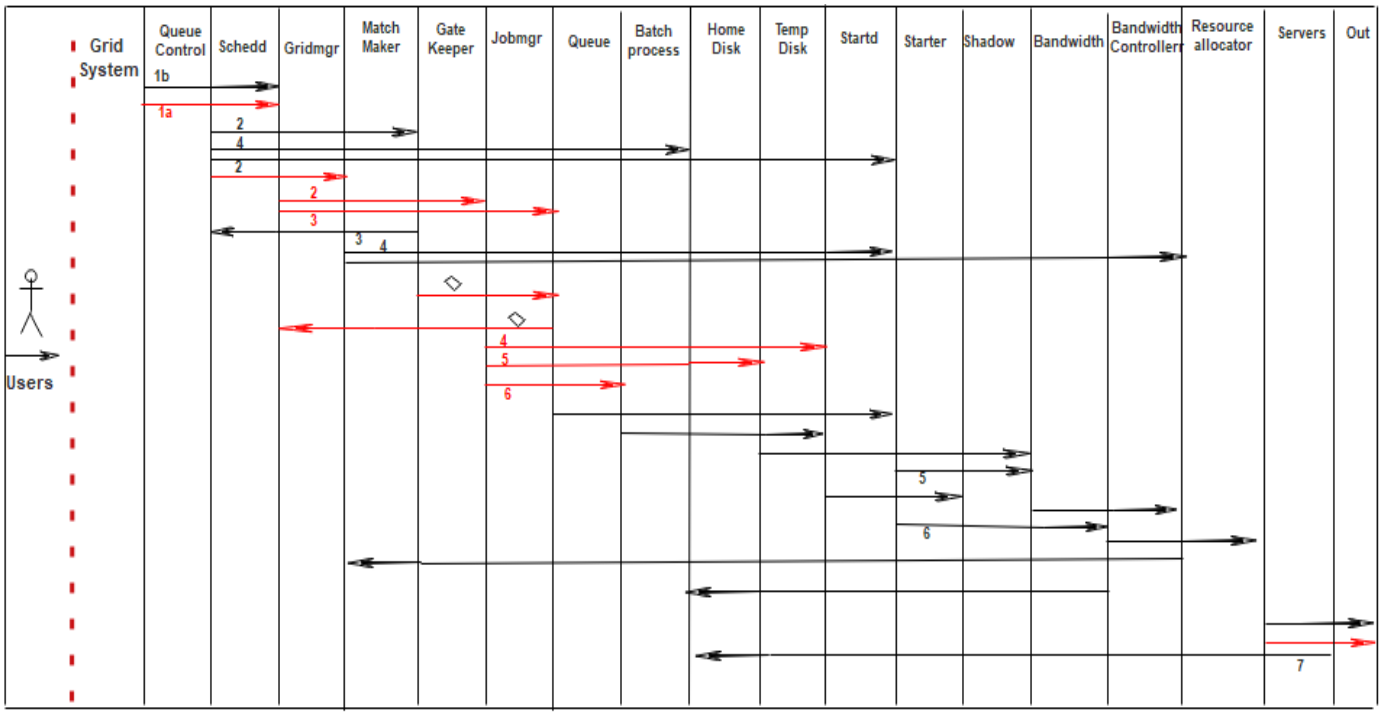


Figure 3 2 Process within Condor-G system

3.3.2 Class Diagram

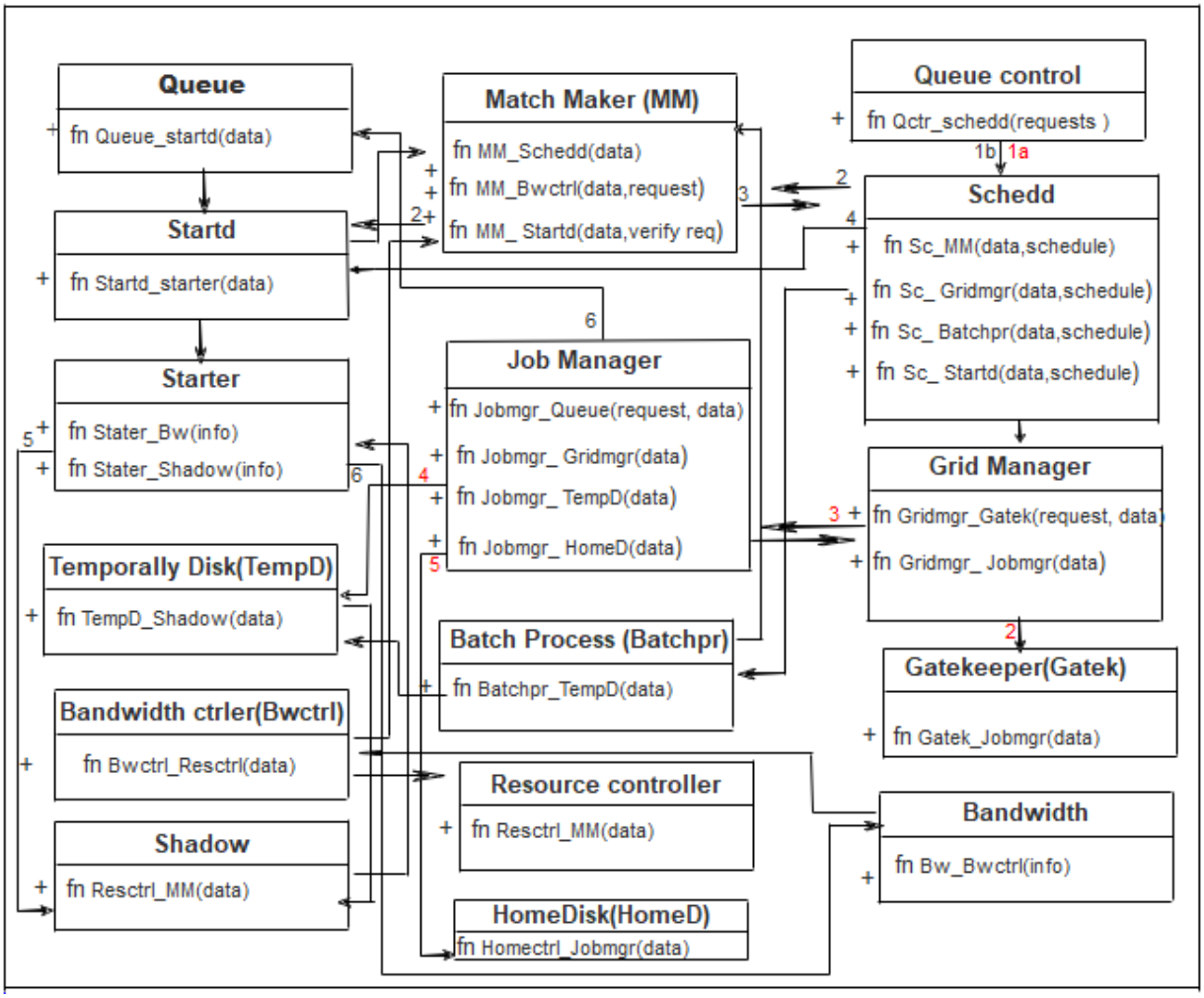


Figure 3 3 Process within condor-G with class diagram

3.4 Documentation

Different documentation for bandwidth allocation to Data centers are available on Internet like Journals, Research papers, Projects done and different book from libraries and electronic libraries.

3.4.1 Observation

The researchers heard about complains from Data center clients specially when accessing data, applications, services or remoting their virtual machine located within the Data center related to slowness. The researchers found that the issue is bottleneck when accessing Data center.

The researcher observed that once the bandwidth is well managed, optimized by allowing bandwidth the whoever need it, performance can be improved.

3. 5 Required hardware and software

3.5.1 Hardware requirement

This project system will require at least the following hardware for the implementation:

1. Computer with 4 GB RAM
2. Core i3 or i5
3. 40GB free space of Hard Disk

Software requirement:

1. Scientific Linux 7 x86 OS
2. Matlab
3. Globilus Condor

3.6 Design process

The proposed new design for bandwidth allocation in Data center is a good idea for solving this issue of bandwidth mismanagement.

The new designed will include Condor-G and will be able to control the allocation of bandwidth where necessary. This design is the improvement of the existing ones for bandwidth allocation but which was not include Condor –G and the objective of this project. As it was done in chapter 5, this scheme performed better than the existing scheme hence our goal was achieved.

3.7 Conclusion

The chapter three of methodology covers the approach to carry out the research, methods of its analysis and the processing for simulating the new system designed for bandwidth allocation in Data center. This new design which use Condor-G will respond to the issue of bottleneck in Data center, also it will speed up. The proposed system is going to be explained in the next chapter.

CHAPTER FOUR: SYSTEM ANALYSIS AND DESIGN

4.1: Introduction

The analysis & design of the dynamic model of datacentre bandwidth allocation using grid by Algorithm, Design, System Level Performance, some flowcharts and algorithms were used as described in chapter three of the research methodology. In this chapter the system of the project, condor-G application is used to allow clients to share available bandwidth to access applications in the Datacentre.

The Client signs the agreement with Data center Manager to allow its (client's) users to have access to the services, applications or data located in the Data center after being authenticated. The signed agreement contains the maximum guarantee bandwidth and the number of users who will access services, application, data, videos and so on.

Once the users of the client was authenticated by Gatekeeper of the System, It allows to access a specific application, service or data stored within Data center. This system will identify the needed capacity of bandwidth to allow this application to be accessible. As the application completes it process, it will automatically release the bandwidth used/occupied by it to other Users. The system will still guarantee the set minimum bandwidth for the client. Often more unused bandwidth (above allocated Max bandwidth to the client) will be provided to the users to complete their processes on or before time.

4.2 System Model

In this system model, a client may have a large number of users Ui (where U represents users and i represents number of users with condition i>0) are accessing different applications/data or services located in the Data center.

The Clients are assigned different amount of bandwidth Bi (where Bandwidth is represented by B while i represents the number of clients) B_1, B_2, B_3 ……B_b . The Bandwidth limit depends

according to the contracts signed between Data center manager and the representative of the Customers or Clients respectively who may have several users U_i .

Data center have different number of devices. These devices are routers, firewalls switches, and servers. Consider R_1, R_2…R_r (where Router is represented by R while i represents the number of routers in Data center), F_1, F_2 …Ff (where Firewall is represented by F while i represents the number of firewalls in Data center) and S_1, S_2, ..S_s (where Switch is represented by S while i represents the number of switches in Data center).

The next layer will be acting as proxy to provide bandwidth to the services or applications requested by different clients using Condor-G.

Let consider C_1, C_2, C_3 .... C_c client, each respectively associated with maximum bandwidth capacity B_1, B_2, B_3,…B_b having users U_1 , U_2.....U_u for each client.

The following diagram illustrate how applications and services are accessible from the Clients to the Data center.

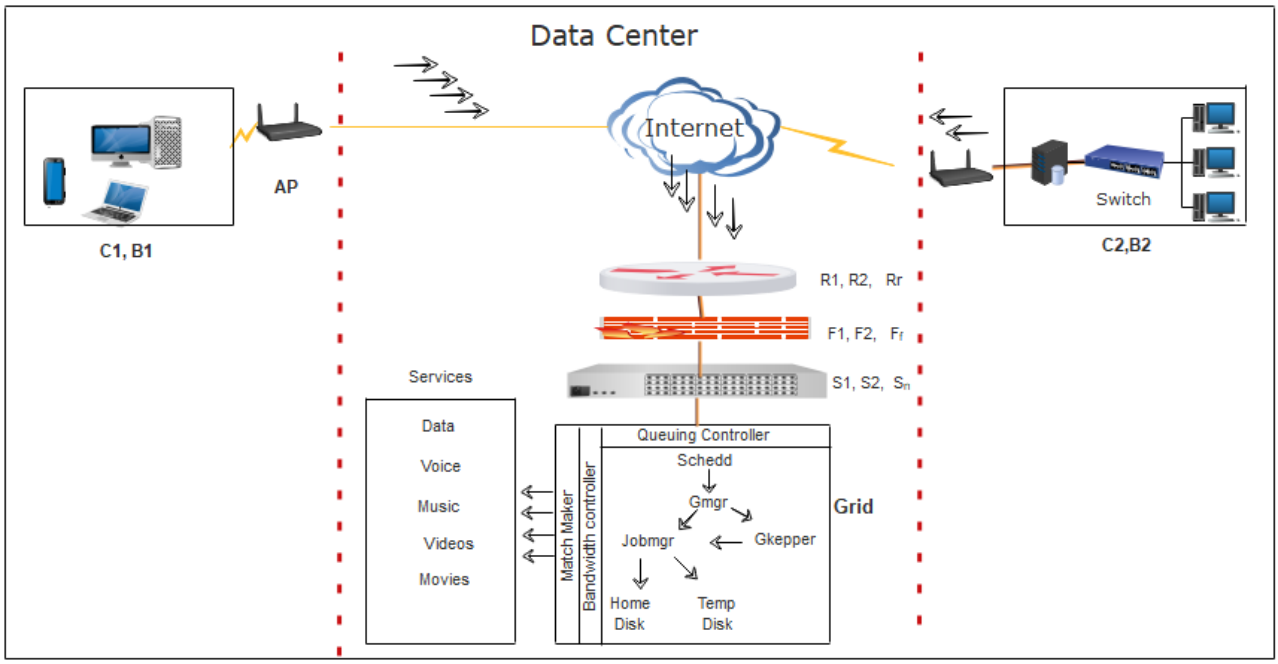


Figure 4 1 Data center traffic flow

The traffic is passing through the different layers of the Data center devices starting from physical routers connect Data center to the Internet and get traffic from outside then send them to the firewalls for filtering and then switches for distribution with limited capacity as signed an agreement between the client and Data center.

The server integrated with Condor-G will allow allocation of bandwidth to applications requested by the users. The next diagram elaborates the condor-G installed inside the Data Center Grid.

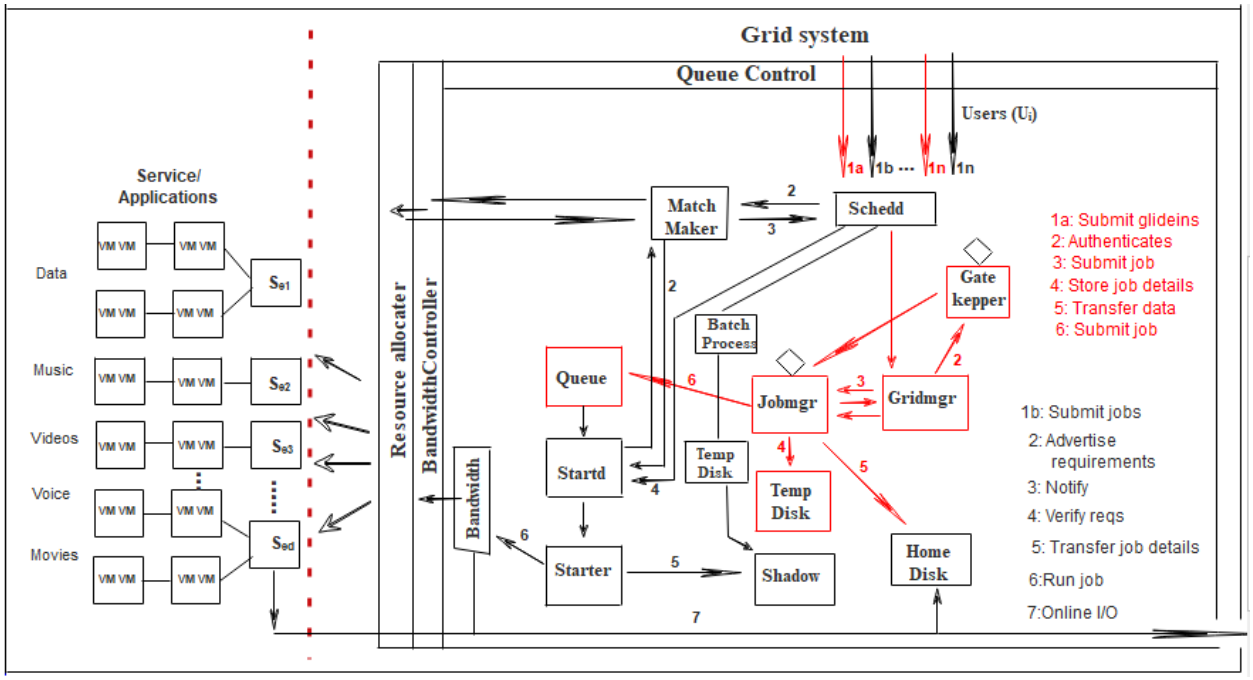


Figure 4 2 Condor-G Application bandwidth allocation

Part I

In the diagram Figure 4.2, the user Ui of client C_1 submits the request to scheduler (Schedd) for accessing an application or any other service located in the datacentre system grid. The schedd transfers the request to *gridmanager* (Gridmgr) *to* process. The *gridmanager* provides details to remote the Queue through the *gatekeeper* that authenticates the users with SSL, Ports mapping, login and so on. The *gatekeeper* use GSI to accept connections, authenticate the client, the guarantee bandwidth for the client and keep the ID track of the client.

When the authentication is done, the client will request a service for being added to the Queue through a jobmanager then reach the Job manager for processing. In parallel all input are added to the cache and also replicated to the Disk or RAID for the future use.
When the job starts running, the gridmanager will be responsible for servicing the job's input and output.

PART II

In parallel (this is called glindIns the Condor System) to activities of PART I the scheduler transfer the incoming data to Match Maker through bandwidth allocator and resource allocator that decides which service Ser $_i$ S_{er1}, S_{er2}, ..S_{ers} (where Service is represented by S_{er} while i represents the number of services in Data center) needs to be utilised and hence where it is located on Server S_{ei} S_{e1}, S_{e2}, ..S_{es} (where Server is represented by S_e while i represents the number of servers in Data center).
The processing continue then starts (Starter) replicate in the mirror server at different location to avoid processing errors and also cross checking results and then finalize submitting the jobs back to Users etc through the Home disk through jobmanager and grid manager.

Activities Diagram

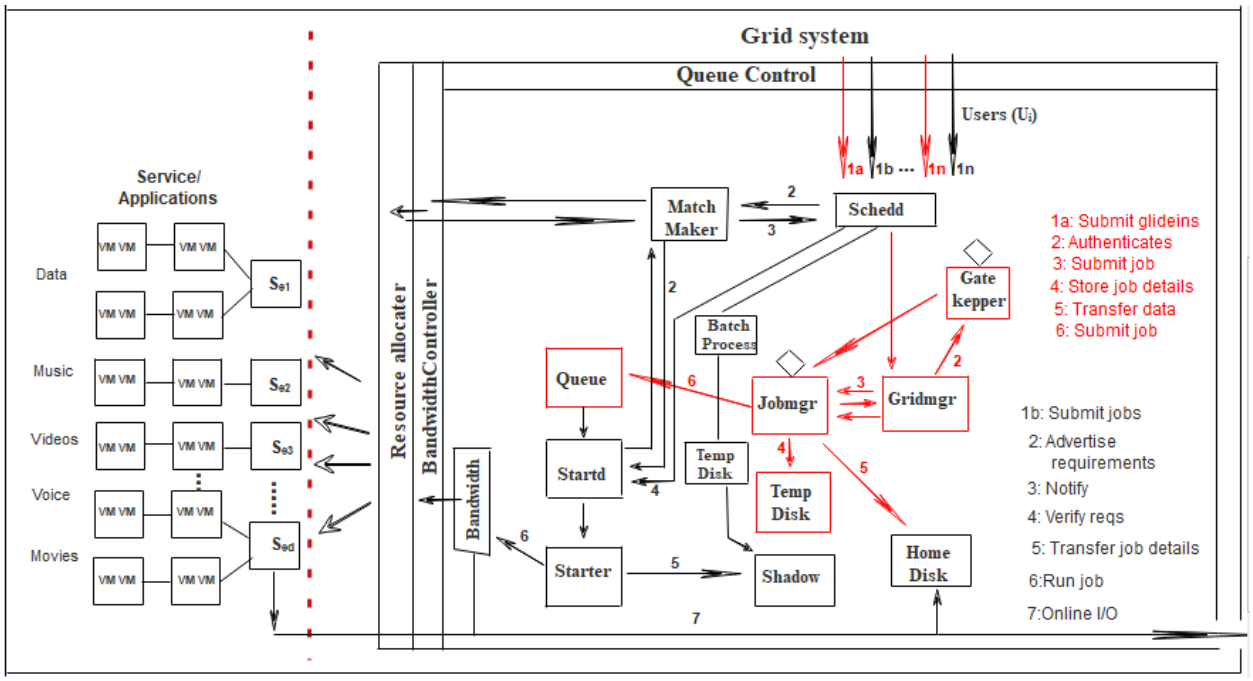


Figure 4 3 Grid system activities

4.3 Results

4.3.1 Queueing by resource allocator

General format A/B/C/N/K……Eq (1)

A is interarrival distribution time

B is the service time distribution

C represent the number of parallel servers

N is the system capacity

K is size of users

For Data center being discussed it has

M/M/es/∞/∞…….

M = exponential type

es = number of parallel servers

So, by conservation Equation

$$L = \lambda\omega \qquad \text{Eq (2)}$$

Where:

L is long-run time average of number of customers in system

λ is the arrival rate of jobs at system

ω is long-run average time spent in the system per customer

Server utilization

It is defined as Proportion of time that server is busy. It is observed by

ρ_o and in defined over the closed time of [0, 1]

So system performance can vary widely for a given value of utilization ρ

Considering system, the utilization depends on the server numbers arrival rate λ and service rate μ and utilization is:

$$\rho = \frac{\lambda}{\mu} < 1$$

For the Data center, the system is a Multiserver Queue like M/M/C/∞/∞

Here, C channels are operating in parallel.

Each of these Channels has identical and independent exponential service time distribution with $\frac{1}{\mu}$ the arrival rate is λ

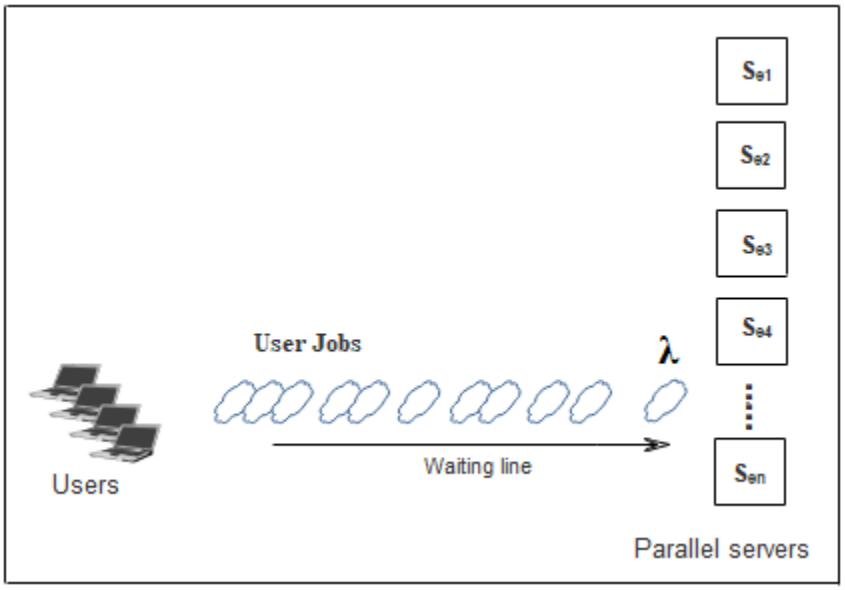


Figure 4 4 Arrival services to the system

The offered load is defined by λ/μ

So if $\lambda > C\mu$ the waiting line grows at rate $(\lambda - C\mu)$ customer per Unit time on average

For M/M/C queue to have statistical equilibrium the offered load must satisfy

$\lambda/\mu < C$ in which case

$$\lambda/(\mu < c) = \rho \quad \text{The server utilization}$$

To define MultiQueued Data center

P_n = at steady state probability of having n customers in the system

λ is the arrival rate

μ is the service rate of one server

ρ is the server utilization

S_n is service time of n^{th} customer

$LQ_{(t)}$ is the number of customers in queue at time t

L = long run time average of customers in the system

ω is the Long run average time spent in system by customer

ω_Q is Long-run average time spent in queue by customers

$$\text{So } \rho = \frac{\lambda}{c\mu} \qquad (1)$$

Where

ρ is the server utilization

λ is the arrival rate

μ is the service rate of one server

C is the number of parallel servers

$$\rho_o = \left\{ \left[\sum_{n=0}^{c=1} \frac{\left(\lambda/\mu\right)^n}{n!} \right] + \left[\left(\frac{\lambda}{\mu}\right)^c \left(\frac{1}{c!}\right) \left(\frac{c\mu}{c\mu - \lambda}\right) \right] \right\} \qquad (2)$$

Where

The probability that servers are busy is:

$$\rho\left(L_{(\infty)} \geq c\right) = \frac{\left(\lambda/\mu\right)^{c} \rho_o}{c!(1-\lambda/c\mu)} = \frac{(c\rho)^{c} \rho_0}{c!(1-\rho)} \quad (3)$$

$$L = c\rho + \frac{\left(c\rho\right)^{c+1} \rho_o}{c\left(c!\right)\left(1-\rho\right)^{2}} = c\rho + \rho\frac{\rho\left(L_{(\infty)} \geq c\right)}{1-\rho} \quad (4)$$

$$\omega = \frac{L}{\lambda} \quad (5)$$

$$\omega Q = \omega - \frac{1}{\mu} \quad (6)$$

$$LQ \quad \lambda\omega Q = \frac{\left(c\rho\right)^{c+1} \rho_o}{c\left(c!\right)\left(1-\rho\right)^{2}} = c\rho + \rho\frac{\rho\left(L_{(\infty)} \geq c\right)}{1-\rho} \quad (7)$$

$$L - LQ \qquad \frac{\lambda}{\mu} = c\rho \quad (8)$$

ALGORIRM FOR THE ENTIRE APPLICATION

Step1: Users make requests

Step2: Scheduler receive the requests

Step3: Scheduler sends requests to Match Maker and Grid Manager

Step4: Grid Manager sends request to Gate keeper and Job Manager

Step5: Gate keeper authenticates requests

```
if (the request is valid) {
  go to Step 6
 }
  else {
        go to Step 17
         }
```

Step6: Gate keeper sends request to Job Manager

Step7: Job Manager keeps requests to Temporary disk and Home disk

Step8: Job Manager sends requests in Queue

Step9: Queue sends requests in the Startd

Step10: Startd sends requests in the Starter

Step11: Starter checks requested bandwidth

Step12: Starter sends bandwidth needed to Bandwidth Controller

Step13: Bandwidth Controller sends the required bandwidth to Resource allocator

Step14: The Resource allocator allocate the required bandwidth

Step15: End

Step16: Resource allocator sends copy to the Home disk for future purpose

Step17: Stop

QUEUING ALGORITM

Step1: Define λ as lamda, μ as myu, C as number of servers, P as P note, p as row, l as L, ω as W, constant1, constant2, sum1, sum2, sum3; LQ as , ….

Step2: Initialisation of all parameters:

(1) $\rho = \frac{\lambda}{c\mu}$

(2) Constant1= $\frac{\lambda}{\mu}$

(3) Constant2= $\frac{C\mu}{C\mu - \lambda}$

(4) L= $C\rho + \frac{\left(math.pow(\rho, p_{\infty})\right)}{1-\rho}$

(5) $\omega = \frac{L}{\lambda}$

(6) $\omega Q = \omega - \frac{1}{\omega}$

(7) $LQ = \lambda \omega Q$

Step5: for(int i=1;i<=C; i++){

for(int n=1;n<=C ;n++){

sum1=sum1+(Math.pow(constant1, n)/factorial(n));

}

sum2=Math.pow(constant1, C)*(1/(factorial(C))*constant2);

sum3=sum1+sum2;

$P_n = \frac{1}{sum3}$

$$\rho = \frac{\lambda}{i * \mu}$$

Print(P_{note})

Print(ρ)

Print (L)

Print (W)

Print (WQ)

Print (LQ)

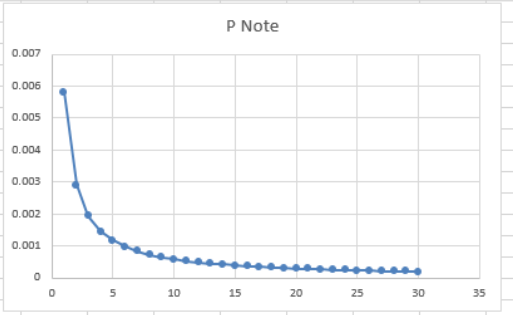
}

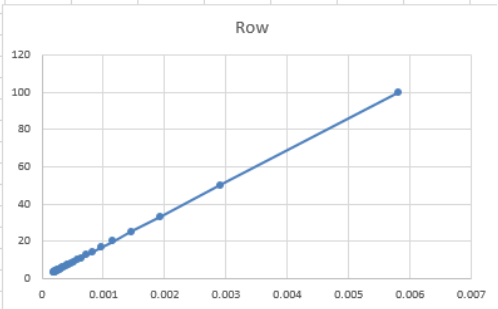
Step6: End

```
run:
value of P Note at 1 is 0.005819767058179175 and row is 100.0
value of P Note at 2 is 0.002909883529089586 and row is 50.0
value of P Note at 3 is 0.0019399223527263912 and row is 33.333333333333336
value of P Note at 4 is 0.0014549417645447938 and row is 25.0
value of P Note at 5 is 0.0011639534116358351 and row is 20.0
value of P Note at 6 is 9.699611763631956E-4 and row is 16.666666666666668
value of P Note at 7 is 8.313952940255959E-4 and row is 14.285714285714286
value of P Note at 8 is 7.274708822723962E-4 and row is 12.5
value of P Note at 9 is 6.466407842421299E-4 and row is 11.11111111111111
value of P Note at 10 is 5.819767058179168E-4 and row is 10.0
value of P Note at 11 is 5.290697325617425E-4 and row is 9.090909090909092
value of P Note at 12 is 4.8498058818159727E-4 and row is 8.333333333333334
value of P Note at 13 is 4.476743890907052E-4 and row is 7.6923076923076925
value of P Note at 14 is 4.1569764701279775E-4 and row is 7.142857142857143
value of P Note at 15 is 3.8798447054527794E-4 and row is 6.666666666666667
value of P Note at 16 is 3.6373544113619806E-4 and row is 6.25
value of P Note at 17 is 3.4233923871642173E-4 and row is 5.882352941176471
value of P Note at 18 is 3.23320392121065E-4 and row is 5.555555555555555
value of P Note at 19 is 3.063035293778511E-4 and row is 5.2631578947368425
value of P Note at 20 is 2.909883529089585E-4 and row is 5.0
value of P Note at 21 is 2.771317646751986E-4 and row is 4.761904761904762
value of P Note at 22 is 2.645348662808714E-4 and row is 4.545454545454546
value of P Note at 23 is 2.530333503556161E-4 and row is 4.3478260869565215
value of P Note at 24 is 2.424902940907989E-4 and row is 4.166666666666667
value of P Note at 25 is 2.3279068232716707E-4 and row is 4.0
value of P Note at 26 is 2.2383719454535306E-4 and row is 3.8461538461538463
value of P Note at 27 is 2.1554692808071045E-4 and row is 3.7037037037037037
value of P Note at 28 is 2.0784882350639947E-4 and row is 3.5714285714285716
value of P Note at 29 is 2.0068162269583405E-4 and row is 3.4482758620689653
value of P Note at 30 is 1.9399223527263965E-4 and row is 3.3333333333333335
value of P Note at 1 is 1.8773442123158683E-4 and row is 100.0
value of P Note at 2 is 1.818677205680998E-4 and row is 50.0
value of P Note at 3 is 1.7635657752058167E-4 and row is 33.333333333333336
value of P Note at 4 is 1.7116961935821168E-4 and row is 25.0
value of P Note at 5 is 1.6627905880511995E-4 and row is 20.0
value of P Note at 6 is 1.6166019606053334E-4 and row is 16.666666666666668
value of P Note at 7 is 1.5729100157241084E-4 and row is 14.285714285714286
value of P Note at 8 is 1.5315176468892638E-4 and row is 12.5
value of P Note at 9 is 1.4922479636356932E-4 and row is 11.11111111111111
value of P Note at 10 is 1.4549417645448012E-4 and row is 10.0
value of P Note at 11 is 1.4194553800437088E-4 and row is 9.090909090909092
value of P Note at 12 is 1.3856588233760017E-4 and row is 8.333333333333334
value of P Note at 13 is 1.35343419957656E-4 and row is 7.6923076923076925
value of P Note at 14 is 1.3226743314043656E-4 and row is 7.142857142857143
value of P Note at 15 is 1.2932815684842688E-4 and row is 6.666666666666667
value of P Note at 16 is 1.2651667517780895E-4 and row is 6.25
value of P Note at 17 is 1.2382483102508962E-4 and row is 5.882352941176471
value of P Note at 18 is 1.2124514704540028E-4 and row is 5.555555555555555
value of P Note at 19 is 1.1877075628937173E-4 and row is 5.2631578947368425
value of P Note at 20 is 1.1639534116358432E-4 and row is 5.0
value of P Note at 21 is 1.1411307957214152E-4 and row is 4.761904761904762
value of P Note at 22 is 1.1191859727267729E-4 and row is 4.545454545454546
value of P Note at 23 is 1.0980692562602301E-4 and row is 4.3478260869565215
value of P Note at 24 is 1.0777346404035595E-4 and row is 4.166666666666667
value of P Note at 25 is 1.058139465123495E-4 and row is 4.0
value of P Note at 26 is 1.0392441175320041E-4 and row is 3.8461538461538463
```

```
value of P Note at 27 is 1.0210117645928463E-4 and row is 3.7037037037037037
value of P Note at 28 is 1.0034081134791768E-4 and row is 3.5714285714285716
value of P Note at 29 is 9.864011963015638E-5 and row is 3.4482758620689653
value of P Note at 30 is 9.699611763632046E-5 and row is 3.3333333333333335
BUILD SUCCESSFUL (total time: 0 seconds)
```

Number of servers	P Note	Row
1	0.005819767	100
2	0.002909884	50
3	0.001939922	33.33333333
4	0.001454942	25
5	0.001163953	20
6	9.70E-04	16.66666667
7	8.31E-04	14.28571429
8	7.27E-04	12.5
9	6.47E-04	11.11111111
10	5.82E-04	10
11	5.29E-04	9.090909091
12	4.85E-04	8.333333333
13	4.48E-04	7.692307692
14	4.16E-04	7.142857143
15	3.88E-04	6.666666667
16	3.64E-04	6.25
17	3.42E-04	5.882352941
18	3.23E-04	5.555555556
19	3.06E-04	5.263157895
20	2.91E-04	5
21	2.77E-04	4.761904762
22	2.65E-04	4.545454545
23	2.53E-04	4.347826087
24	2.42E-04	4.166666667
25	2.33E-04	4
26	2.24E-04	3.846153846
27	2.16E-04	3.703703704
28	2.08E-04	3.571428571
29	2.01E-04	3.448275862
30	1.94E-04	3.333333333





CHAP V: CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this project, the main objective of dynamic bandwidth provisioning for datacenter was to solve the issue of bottleneck when users are accessing data, voice, application, services or virtual machine located in the Data center.

With Condor-G, available bandwidth will be managed in such way that only data, application or service which need bandwidth will be provided if not no bandwidth will be provided.

By optimizing the available bandwidth like unused bandwidth is allocated to the user who need more than what the client has signed for to complete fast as usual, prioritized some services like videoconference, CCTV camera for surveillance, and so on, the performance will be achieved and issue will be solve for the users and Data center managers

5.2 Recommendation

Data centers have many issue related to bandwidth, security, hardware level and Grid has capability to solve many issues.

As recommendation to the next researchers, after Data center has signed the contract with Client, the users may exceed the bandwidth guarantee and this bandwidth has to be paid so using improvement of this project can help data center to earn many.

Bandwidth allocation using Condor-G may be extended to other Data centers connected to the main so that users can access data, services or application wherever he/she want. This can be used also for redundancy of Data centers.

Using Condor-G by prioritized some services and applications such uTorrent for downloading, updates to be done night time can also improve the performance.

References:

[1] "Towards Application-aware In-network Bandwidth Management in Data Centers" Renhai Xu, Wenxin Li, Keqiu Li, Heng Qi 2016 IEEE TrustCom/BigDataSE/ISPA
(Page 2211)

[2] https://techterms.com/definition/grid_computing [50]

[3] I.Foster, C.Kesselman, C.Lee, R.Lindell, K.Nahrstedt, A.Roy. "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", Intl Workshop on Quality of Service, 1999.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards Predictable Datacenter Networks. In *ACM SIGCOMM*, 2011.

[5] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *CoNEXT*. ACM, 2010.

[6] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, C. Kim, and A. Greenberg. EyeQ: Practical Network Performance Isolation at the Edge. In *USENIX NSDI*, 2013.

[7] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese. NetShare: Virtualizing Data Center Networks across Services. *UCSD TR*, 2010.

[8] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. FairCloud: Sharing the Network in Cloud Computing. In *ACM SIGCOMM*, 2012.

[9] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *USENIX WIOV*, 2011.

[10] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the Data Center Network. In *Usenix NSDI*, 2011.

[11] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: sharing the network in cloud computing," in *Proceedings of ACM SIGCOMM*, Helsinki, Finland, 2012.

[12] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese, "Netshare: Virtualizing data center networks across services," niversity of California, San Diego, Tech. Rep., 2010.

[13] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: practical work-conserving bandwidth guarantees for cloud computing," in *Proceedings of ACM SIGCOMM*, 2013.

[14] J. Guo, F. Liu, X. Huang, J. C.S.Lui, M. Hu, Q. Gao, and H. Jin, "On efficient bandwidth allocation for traffic variability in datacenters," in *Proceedings of IEEE INFOCOM*, Toronto, Canada, 2014.

[15] B. Briscoe, "Flow rate fairness: Dismantling a religion," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 63–74, 2007.

[16] The Grid 2 Blueprint for a New Computing Infrastructure (The Elsevier Series in Grid Computing) Second Edition Edited by Ian Foster Argonne National Laboratory and the University of Chicago, Carl Kesselman Information Sciences Institute, University of Southern California

[17] https://searchnetworking.techtarget.com/feature/What-are-data-center-class-switches

[18] https://en.wikipedia.org/wiki/Shannon%E2%80%93Hartley_theorem

[19] Telecommunications Essentials, Second Edition: The Complete Global Source, 2nd Edition, By Kitty Wilson Jarrett and Lillian Goleniewski, Oct 12, 2007

[20] https://www.lifewire.com/introduction-to-network-cables-817868

[21] https://www.ctctechnologies.com/network-cabling-types/

[22] https://www.thegeekstuff.com/2010/08/raid-levels-tutorial

[23] https://www.webopedia.com/quick_ref/servers.asp

[24] Cloud Architecture and Datacenter Design, in Distributed computing: Clusters, Grids and Clouds, All rights reserved by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010

[25] A Scalable, Commodity Data Center Network Architecture by Mohammad Al-Fares malfares@cs.ucsd.edu, Alexander Loukissas aloukiss@cs.ucsd.edu, *SIGCOMM'08,* August 17–22, 2008, Seattle, Washington, USA. Copyright 2008

[26] J. Guo, F. Liu, D. Zeng, J. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proceedings of IEEE INFOCOM*, Turin, Italy, 2013.

[27] Optical fibres, cables and systems, International Telecommunication Union Place des Nations CH-1211 Geneva 20 Switzerland, Mr. Yoichi Maeda in Switzerland Geneva, 2009

[28] Data Center Network Virtualization: A Survey by Md. Faizul Bari_, Raouf Boutaba_, Rafael Esteves_*y*, Lisandro Zambenedetti Granville*y*, Institute of Informatics Federal University of Rio Grande do Sul Av. Bento Gonc¸alves, 9500 Porto Alegre, Brazil

[29] Journal of Grid and Distributed Computing by Indu Gandotra, Pawanesh Abrol, Pooja Gupta, Rohit Up pal and Sandeep Singh, Volume 1, Issue 1, 2011

[30]Future Generation Computer Systems 25(2009)599-616www.elsevier.com/locate/fgcs the University of Melbourne, Australia

[31] C Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu. Loud “Computing and Grid Computing 360 Degree Compared“

[32] Comparative study between Cluster, Grid, Utility, Cloud and Autonomic computing by Samah Mawia Ibrahim Omer, Amin Babiker, A.Mustafa Fatema, Abdallah Elmahdi Alghali

e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 9, Issue 6 Ver. III (Nov–Dec. 2014)

[33] Grid Computing Architecture and Benefits by Shruti N. Pardeshi, Chitra Patil, Snehal Dhumale International Journal of Scientific and Research Publications, Volume 3, Issue 8, August 2013 ISSN 2250-3153

[34] Foster, C. Kesselman, J. M. Nick and S. Tuecke, “The physiology of the Grid:An open grid services architecture for distributed systems integration”, Grid Forum white paper,2003.