YAŞAR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

EFFICIENT RESOURCE MANAGEMENT FRAMEWORK IN CLOUD COMPUTING

Bashir Yusuf Bichi

Thesis Advisor: Asst. Prof. Dr. Tuncay ERCAN

Department of Computer Engineering

Presentation Date: 20/6/2014

Bornova-İZMİR
2014

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Assist. Prof. Dr. Tuncay ERCAN

(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Prof. Dr. Mustafa Gündüzalp

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Assist. Prof. Dr Korhan KARABULUT

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Prof. Dr. Behzat GÜRKAN

ABSTRACT

EFFICIENT RESOURCE MANAGEMENT FRAMEWORK IN CLOUD COMPUTING

Bashir Yusuf Bichi
M.Sc. in Computer Engineering
Supervisor: Asst. Prof. Dr. Tuncay ERCAN
June 2014

Cloud Computing is a kind of public utility service which gives the client room to focus on his work without focusing on installation and maintenance of the important devices in their system as they are installed and maintained by the Cloud Service Providers. Cloud computing is meant to be scalable, and enhance the quality of service (QoS), cost effective and also simplified user interface so that the customer can appreciate the idea behind cloud computing. User requests in Cloud Computing in dealing with resource allocation are examined in Queue structures. In order to solve the resource allocation problems, different queue models and resource allocation optimizations are used. In this thesis, we focused on mathematical formulations to show how throughput and time delay, level of occupancy or utilization and the response time may vary between a single server system and a virtualized multiple server system in a cloud-computing environment. Virtualization technology is employed in the field of cloud computing in order to complement the activities of physical server system. We tried to develop a new framework based on Max-Min algorithm which aims at distributing load to a set of virtual resource system using various balancing techniques and show the results with a MATLAB simulation

ÖZET
BULUT BİLİŞİMDE ETKİN KAYNAK YÖNETİM YAPISI

Bashir Yusuf Bichi
Bilgisayar Mühendisliği Yüksek Lisans
Danışman: Yard.Doç.Dr.Tuncay ERCAN
Haziran 2014

Bulut Bilişim kullanıcı sistemlerindeki önemli cihazlarının tesis ve bakımının Bulut Hizmet Sağlayıcıları tarafından karşılandığı ve doğal olarak kullanıcıların bu işlevleri düşünmeksizin kendi işlerine konsantre olabilmelerine yardımcı olan herkesin faydalanabileceği bir hizmet şeklidir. Bulut Bilişim ölçeklenebilir olması, yüksek hizmet kaliteli hizmet sunabilmesi, maliyet etkin ve basitleştirilmiş kullanıcı ara yüzleri gibi özellikleriyle müşterilerin takdir ettiği bir sistem sunar. Bulut Bilişimde kullanıcı istekleri ve ortak sistem kaynakları arasındaki ilişki kuyruk yapıları içinde incelenmektedir. Kaynak tahsis problemlerini çözebilmek için farklı kuyruk modelleri ve kaynak tahsis iyileştirmeleri kullanılmaktadır. Bu tez çalışmasında Bulut Bilişim ortamında tek veya sanallaştırılmış çoklu sunucu içeren sistemlerde kullanım ve doluluk düzeyinin, verimlilik ve zaman gecikmesinin, tepki süresinin nasıl değiştiğine ilişkin matematiksel formüller üzerinde durulmuştur. Sanallaştırma teknolojisi Bulut Bilişim alanında fiziksel sunucu sisteminin faaliyetlerini tamamlamak için kullanıldığı için, çalışmamızda farklı dengeleme teknikleri kullanan sanal kaynak sistemi için uygun yük dağıtımını amaçlayan “Max-Min” algoritmasına dayalı etkin bir yapı geliştirmeye çalışılmış ve sonuçları simülasyonla gösterilmiştir.

ACKNOWLEDGMENT

It would not have been possible to write this Master's thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here. Above all I would like to appreciate; the effort of my supervisor Assist Prof. Dr. Tuncay Ercan who always gave me his support. He provided me with excellent suggestion and feedback on my thesis, pointed out my mistakes.

I would like to thank my family for their personal support and great patience at all times. My course mates for their wonderful suggestion and friendship. I would like to acknowledge the academic support of Yaşar University and its staff, particularly Computer Engineering Department. My gratitude also goes to the Head of Department, Prof. Dr. Ahmet Koltuksuz for support and assistance since the start of my postgraduate work in 2012. Last but not the least; I would like to thank my family: my parent Maimuna Sulaiman and Yusuf M Bichi, for supporting me since I was a child.

TEXT OF OATH

I declare and honestly confirm that my study, titled Efficient Management Framework in Cloud Computing and presented as a Master's Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions, that all sources from which I have benefited are listed in the bibliography, and that I have benefited from these sources by means of making references.

TABLE CONTENT

Contents

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF SYMBOLS AND ABBREVIATIONS

Symbols	Explanations
λ	Arrival rate of a request
μ	Service rate
R	Expected number of Request
T_r	Mean reasons time
Q	Expected number of request in Queue
Tq	Expected time spend on the queue
Th	Throughput
P_{busy}	Busy time of the system
ρ	Utilization rate of a system
v_j	Virtual machine resource
t_i	Task to be executed
s_i	Data file size of a task t_i
q_i	Processing power of a task t_i
EET	Expected execution time
CT	Completion time

Chapter One: Introduction

1.0 Evolution of Cloud Computing

In recent years cloud computing has become a subject of interest as it is a type of computing that heavily relies on sharing of computing resources rather than having local servers or personal devices to handle application in-house. The goal is to apply some traditional supercomputing or high-performance computing power, to perform computations, in consumer oriented applications, to deliver personalized information, to provide data storage and so on.

Cloud computing is described as the hardware or software service delivery over the internet. Cloud computing has been a topic of discussion in recent years due to the need for resource management and efficiency. Cloud computing evolved through number of phases such as grid and utility computing, application service provision (ASP) and software as a service (SaaS) as shown in fig. 1.1, cloud computing can also be viewed as a kind of innovation in different way such as in technological perspective; which is the advancement of computing, using virtualization concept for efficient use of hardware. Another perspective of cloud computing can be seen from IT deployment in which cloud computing resources and applications are provided in a way that is different from the traditional approach [1].

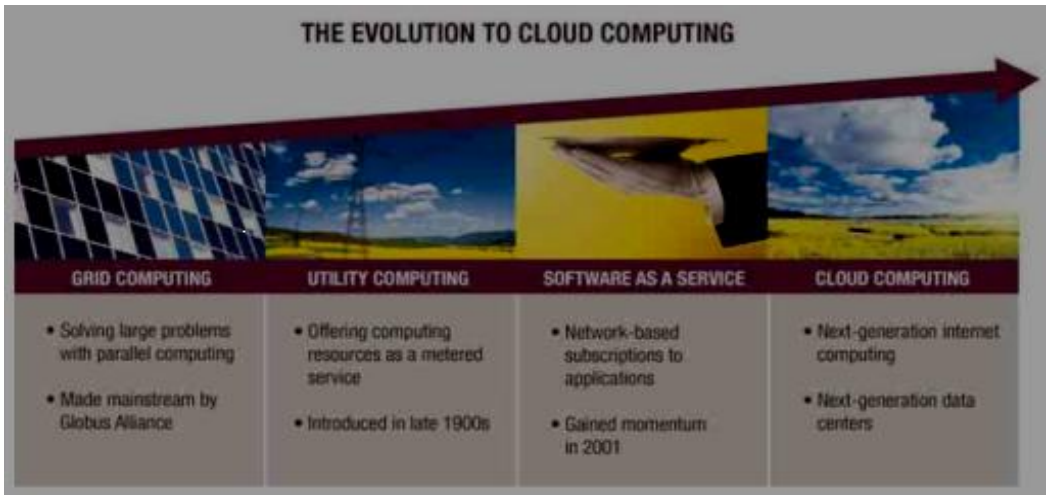


Fig. 1.1 Evolution of Cloud Computing [44]

1.1 Cloud computing as a new Paradigm.

In the early days of computer technology, the mainframe computer was physically very large, centralized computing platform with terminals used by end-users. These terminals could be compared to thin client devices in today's industry and the mainframe as the centralized cloud computing platform. This centralized mainframe held all of the computing power (CPU), memory, and storage systems managed by a small staff for shared use by a massive number of users [27].

Today, cloud computing is seen as a new paradigm, however some researchers suggest that one needs to reflect back through history i.e. the context of computing history. The term cloud computing became popular in 2008, however its practice in which computing functions through network that are provided remotely dated back to the mainframe time-sharing systems, back in the 1960s. Utility computing became very challenging as effort to create large-scale computing utilities faces constrains like in the telecommunications networks were the capacity is less, however in the 1990s, the constrain on network capacity gradually became history, as companies invested in high-capacity fiber-optic networks which see the internet as a medium for rapid information exchange. Many companies involve in the provision of application remotely over the internet, this companies are called application service providers (ASP) [28]. This idea of providing application remotely over the internet was later renamed as the cloud computing. Due to increase in technologies such as JAVA, PHP and so on, which make it possible to develop more elaborate and interactive websites that give possibilities to find many multimedia websites, online transactions and many applications that can be deployed in the internet which includes; communication platforms, social networks, office application etc [14]. Fig.1.1 below shows how cloud computing evolve from some computing paradigm.

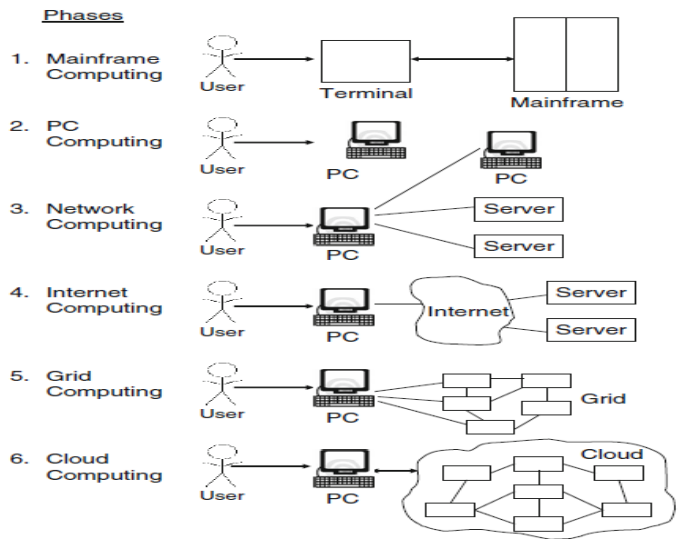


Fig. 1.1 cloud computing paradigm [14]

This lead to the conclusion that cloud computing is a return to the original mainframe computing, however there exits some differences in the two paradigms. In mainframe computing finite computing power is experienced while cloud computing offers infinite computing power, terminals are used as interface in mainframe computing while cloud computing uses powerful PCs, are employed.

This deployment concept is refered to as Software-as-a-Service, which became popular in the early 2000. It also give room to the development of similar deployments strategies such as the development of hardware resources (e.g. computing power and storage) which lead to the establishment of what is called grid computing. According to Computer Weekly; cloud computing experience its first milestone with salesforce.com in 1999, Amazon web services in 2002, Google, and so on [51].

1.2 Pros and Cons of Cloud Computing

Cloud computing advantages include;

- Scalability/Resource flexibility
- Better hardware management

- Easier to share content e.g. social networking
- Ability to access application anywhere, and at all time
- Save money/cost effective as customer pay for only what he uses.

Despite the remarkable advantages experienced in the cloud computing there are some setbacks as well, including;

- Lack of trust between companies and storage providers.
- Depending on third-party; as the provider may possibly shutdown service which could make it hard to retrieve the data.
- Peripherals such as printers, scanners may not work.
- Relies 100% on network connection as it requires constant internet connection, and at high speed.

1.3 Cloud Computing Companies.

Today there are various cloud computing companies with the aim of providing their clients with reliable and efficient services, some of these companies include [2];

1.3.1 Apple: The Apple iCloud.

Apple offers its first cloud services with what is known as iTunes virtual music store which offers millions of songs for download through web-base storage devices apart from music iTunes. It also laid the foundation for scalable e-commerce, high-bandwidth download transactions and user device independence.

1.3.2 Microsoft: Windows Azure Platform.

Windows Azure is a Microsoft platform which allows developers to move their applications to the cloud. The Windows Azure provides operating-system support for .NET applications and SQL server (cloud based) known as SQL Azure. Windows Azure is scalable to the developers, i.e. the developers' application grows in terms of users, processor demands, and/or disk storage, the

Windows Azure environment grows to meet such needs. Microsoft also provides what is known as Microsoft SharePoint online service in which contents and business tools are allowed to move into the cloud. It also makes office applications available over the cloud.

1.3.3 Amazon: Amazon Web Services (AWS).

Amazon is among the top e-commerce companies, Amazon grew its infrastructure to give the developers chance to use the Amazon network resources. The company releases what is known as Amazon Web Services (AWS) which allows companies to host their systems. AWS processes thousands of web-based requests for companies in almost every second. Some components of the Amazon Web Services (AWS) include; Amazon Elastic Compute Cloud (EC2), Amazon Simple Storage System, Amazon Elastic Block Store, Amazon SimpleDB, Amazon Relational Database Service, Amazon Cloudfront among others.

1.3.4 Salesforce.com

Salesforece.com founded in 1999 is seen today as the first among such companies to lunch a large scale SaaS solution. Salesforce.com gives its clients (i.e. salesperson) the chance to spend at least ¾ of his/her time to non sale tasks such as calendar management, contract management, contact management etc. The salesforce.com's customer relationship management is categorized into sale cloud, service cloud, data cloud, collaboration cloud and custom cloud.

1.3.5 Google:

Google provides to its customers a wide range of activities. It uses a private cloud that delivers variety of services to its users which include; document application, email access, maps, text translators and many more[3]. Google provide its users with an App Engine which allows them to run their applications on Google's infrastructure. The App Engine applications are easy to build, easy to maintain, and can easily scale up to meet user's demand, the App Engine platform support programming languages like, Python, Java, PHP and so on.

1.4 Benefits of Cloud Computing

Traditional business application is somewhat very expensive as the amount and variety of hardware and software required to run a particular application is very high, as one need a team of experts to install, configure, test, run, secure and update such application.

Cloud computing offers some significant benefits to its customers, according to salesforce.com a client is free from hardware and software management as it is the responsibility of the vendor. The clients share the resources, i.e. no permanent owner to a given resource due to its utility like structure, you only pay for what you use, upgrading is automatic and scaling up or down is easy. Could computing applications cost less and they can only be deployed within days or a week, a client only needs a browser to log in and start using.

Barrier [4], outline some of cloud computing which are also seen as characteristic of efficient cloud computing. Some of this includes;

- Resource pooling: the cloud service providers develop what is called resources pool which allows multiple users to use the resource (i.e. multitenant usage) at the same time.
- Rapid Elasticity: the cloud computing system adds resources by scaling the system up or down. Scaling is the ability of a cloud computing system to add resources when needed or to reduce it when not in use.
- On-demand self-service: this means resources can be provisioned by the client without the consent of the cloud computing service provider.
- Measured service: the client use of resources is measured, audited and reported back to the client by the service providers.
- Lower cost: significant cost reduction is achieved or experienced by using cloud computing network as it operate in higher efficiency and with greater utilization.

- Broad network access: this means client can gain access to cloud resources over any available network using standard method in a way that is platform independent.

1.5 Challenges in Cloud Computing

Cloud computing makes life easy to its users due to its promising capabilities, however it does have some challenges, some of which includes [2, 3,27];

- Compliance issues: different geographic area employs different law and policies, which implies that the cloud must accommodate multiple compliance regimes. For instance Sarbanes-Oxley Act (SOX) in the US (act passed by the US congress) and data protection directives in the EU are example of compliance issue affecting cloud computing, this is because the EU has legislative backing for data protection across all its stake holders, but the US data protection policy is different and can vary from one State to other.
- Security and Privacy issue: cloud users are concerned to put their data, and running on a remote system, as security challenges such as data loss, phishing, and botnet may pose serious threats to such users. The multi-tenancy and pooled nature of cloud computing also poses more and serious challenges, therefore to ensure data privacy, additional security method are needed to be put in place. Some suggest that private encryption, VLANs, firewalls and local storage of sensitive data are needed.
- Continuously evolving: as cloud computing continue to grow, both the user requirement and the requirement for the interfaces, networking and storage continue to evolve, which means the cloud is continuously dynamic, particularly the public cloud.
- Procurement and budgeting for cloud services is a challenge to some commercial and government organizations. Existing procurement policies may need to be adapted.

1.6 Cloud Computing Technologies

According to NIST, cloud computing is a model for enabling convenient on demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing technology was actualized as a result of advancement in technologies with regard to the hardware, internet technology, distributed computing and system management. The figure below shows the technologies that result in cloud computing [13].

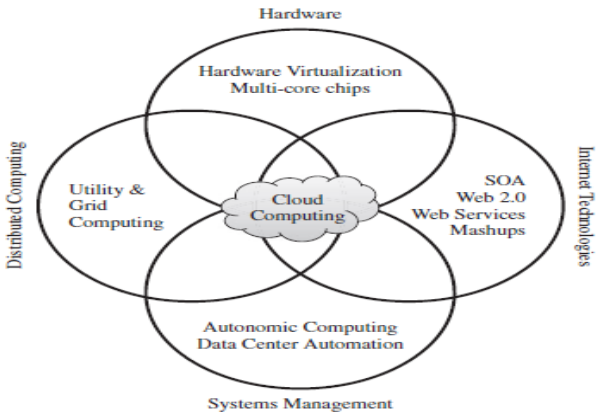


Fig. 1.3 cloud computing technologies [13]

1.6.1 Internet Technologies

- Web Services: The web services is seen as a server that glued applications that are running on different messaging product platforms together, enable information availability from one application to the other, and enable applications to be available over the internet. The web services standards are created above some ubiquitous technologies like HTTP, XML among other technologies, which provide a mechanism for service delivery, and also make it ideal for the implementation of service-oriented architecture (SOA).
- Service Oriented Architecture (SOA): The SOA is a collection of services that communicate with each other, the communication can be either simple data passing or involving two services coordinating some activities. It is an

application development methodology that is used by developers to create a solution through integrating one or more web services [2].

- Web 2.0 and Mashup: The Web 2.0 describes the set of tools and websites that gives the user a room to publish contents to the web without the direct use of HTML. This is because the tools and sites build the HTML documents for the user behind the scene and then upload the documents to a web server. Example of such applications and sites include YouTube, Twitter, Facebook etc. Web 2.0 is a concept that references to the use of web technology and web design to enhance creativity, information sharing, and collaboration among users [14]. The Mashup is a collection of services joined to create an overall solution. Many companies need a variety of SaaS solutions; some developers categorized mashup as web-base or server-base. The web-base mashup allows the user's browser through javascript to combine various content sources to create a unified display, while the server-base mashups runs a particular application that combines the data on the server [2].

1.6.2 Distributed Computing

- Grid Computing: it is often confused as cloud computing, grid computing is a type of parallel and distributed computing that gives the capabilities of sharing, selection and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost and users' quality-of-service requirement [11]. The key aspect of grid vision realization is building standard web services-based protocols that allow distributed resources to be discovered, access, allocated, monitored, accounted for and generally manage as a single virtual system [13]. In short, grid computing is to use the resources of different computers in a network to work on a single problem at the same time. For example the Search for Extraterrestrial Intelligent (SETI) project.
- Utility Computing: computing and storage resources used by the client are metered in a given way just like water, electricity, gas and telephony. The customers have the capability of using the utility services whenever they decide. The idea of utility computing was actualized in 1960 but due to inadequate technologies and devices, the idea eventually faded [11].

1.6.3 Hardware Virtualization and Multiple Chips:

Virtualization is aimed at building large data centers for the purpose of serving many users and to host many desperate applications seeking services in the cloud. The idea of the virtualization of resources is to improve sharing and utilization of the computer systems. These resources include; processors, memory, and I/O devices. The hardware virtualization allows running of multiple operating systems and software stacks on a single physical platform [13]. For instance, the Virtual Machine Monitor (VMM) or hypervisor help to mediate access to a physical hardware by presenting each guest's operating system a virtual machine (VM). The VM is a virtual platform interface. Example of such VMM includes VMware, Xen, and KVM among others. Other technologies like the multi-core chips and others contribute a lot to the increase adoption of virtualized server system. Virtualization supports some features which include [34];

- Flexibility and adaptability: Infrastructures are flexible and adaptable to more resources. This means that virtualization technology can simulate and adapt to different platforms and manage the resources.
- Infrastructure and location independency: virtualization technique provides platform independency and its services can be accessed independent of the location of the user and the resources.
- Ease of use: users can easily develop new applications, thereby reducing the overhead of controlling the system.

There are many virtualization technologies; each of which aims at providing an abstraction layer between the virtualized resources and physical resource. These technologies include; storage virtualization, database virtualization, memory virtualization and network virtualization. In cloud computing, virtualization aim at providing a high-level of abstraction and self-service interface for provisioning, control and management of virtualized resources that are hosted by various virtualization technologies. Server virtualization can be of two types, which are; full virtualization and para-virtualization. In full virtualization, a VMM or hypervisor serves as a dividing layer between the server hardware and the virtual server. However, full virtualization is unaware of

virtualized environment. The hypervisor mediates access to the server's hardware and its peripherals example is the VMware, para-virtualization is a modification of full virtualization as it is aware of the virtual environment. The example of the para-virtualization is Citri Xen [34]

1.6.4 Autonomic Computing

This gives researchers room to improve systems capabilities by decreasing human involvement in their operation; i.e., system should be able to manage themselves with high-level of guidance from human. Autonomic systems exhibit the abilities like self monitoring, self repairing, and self optimizing by constantly sensing themselves and their performance [15]. The data center of a given cloud computing providers that are large, must provide an efficient way of managing their system. Thus, autonomous system technology helps in providing software technologies for cloud data center automation that performs tasks like [13];

- Management of service levels of running application.
- Management of data center capacity.
- Proactive disaster recovery.
- Automation of VM provisioning.

1.7 Cloud Computing Components

Cloud computing is made up of some components which are described as cloud computing solution. This components include; clients, datacenter, and distributed servers [6, 11]. Each of these components or elements plays a specific role in delivering a cloud based activity. Fig. 1.4 below gives an overview of the essential components that are used in cloud computing environment;

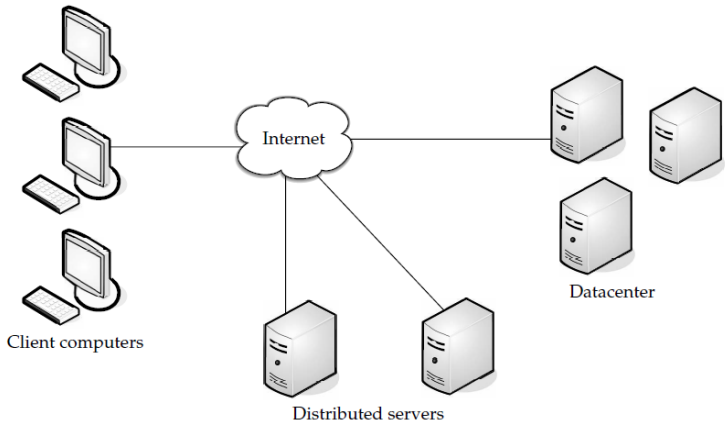


Fig 1.4 Cloud Components [16]

1.7.1 Client

The client is described as a device in which the user (end user) interacts with, in order to manage information over the cloud. It is an interface between the cloud to the user through web browsers and thin computing [17]. The client devices include computers, mobile phones, PDA and so on. Client is further categorized into three main categories which are; Mobile client, e.g., PDA or Smartphone, Thin client e.g. computers with no internal hard drive and, Thick client, e.g., regular computers that uses browsers to connect to the cloud.

1.7.2 Datacenter

The datacenter is a group of services which holds subscribers application. They are infrastructure that provides customers with effective services [16]. The datacenter serves as resource pool, where resources are located. It allows for multiple clients to share common resources otherwise known as multi-tenancy [5].

1.7.3 Distributed Servers

The servers may not necessarily be close to each other geographically, however, the subscriber will assumed that the servers are close to each other. An instance of such distribution is the Amazon cloud service. Amazon cloud solutions are put in different servers all over the world. The main purpose of this

is to make sure that whenever there is failure in one site, the service will still be accessible by client through other sites [16].

1.8 Cloud Computing Deployment Model

Cloud computing is classified according to who owns and manages the cloud, Therefore it consists of four kind of models (Deployment models) as shown in fig. 1.5 below;

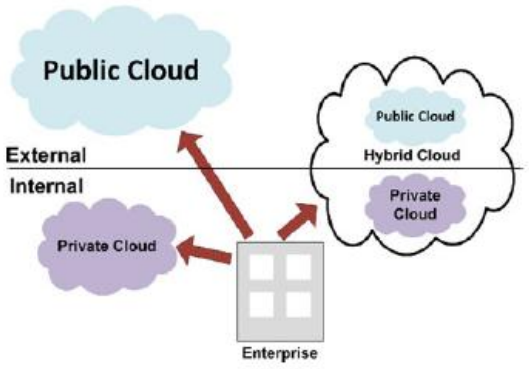


Fig. 1.5 Cloud Deployment model [handbook of cloud computing]

1.8.1 Public Cloud

Public or external cloud is the most common cloud computing model where services are made available to the public in a pay-as-you-go approach [14]. Service providers make resources such as application and storage, available to the public over the internet. Public cloud computing includes; IBM's Blue Cloud, Sun Cloud, Amazon Elastic Compute Cloud (EC2), Google AppEngine, Windows Azure etc. Multi-tenancy is a key characteristic of public cloud services as shown in fig. 1.6 below.

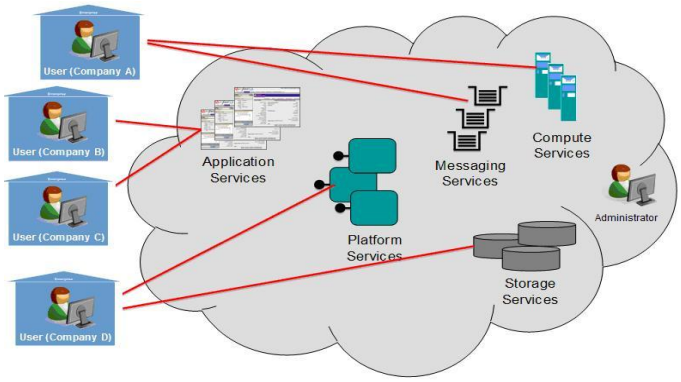


Fig. 1.6 Public cloud deployment model [www.ibm.com]

1.8.2 Private Cloud

Organizations can own, maintain and operate their cloud, the operation may be in-house or with a third party on the premises [2]. Private cloud derives efficiency, standardization and best practices while retaining greater customization and control within the organization. All resources in private cloud environment are local and dedicated, the cloud management is also local [19]. An instance of private cloud deployment is that of eBay, HP and so on.

1.8.3 Hybrid Cloud

The hybrid cloud combine multiple cloud models (i.e. public and private cloud) all cloud models retain their unique identity but they are bounded together as a unit. Hybrid cloud has the ability to allow data and/or application to be move from one cloud to another through their interfaces. Major vendors of hybrid cloud deployment include HP, IBM, Oracle, VMware etc. They create appropriate plans to leverage a mixed environment [11].

1.8.4 Community Cloud

The community cloud aims at serving common functions within a specific organization or among several organizations that shares common concerns such as; mission, policies, security, etc. Community cloud in some extent overlaps with

Grid, as several organizations in private community share cloud infrastructure [11].

1.9 Cloud Services

In cloud computing services refer to the concept of using reusable components across a vendor's network known as "as-a-service". Cloud computing consist of three types of computing services. These services are delivered remotely to the client through the internet [20]. The client using these services pays a service fee to the service providers in order to gain access to their systems. Cloud services posses some traits which includes;

- Low barriers to entry, making them available to small business
- Multi-tenancy: resources are shared among clients
- Device independence, client can access the system using different hardware.

The three known services offered by cloud computing consist of software-as-a-service, platform-as-a-service, and infrastructure-as-a-service as shown in fig. 1.7 below.

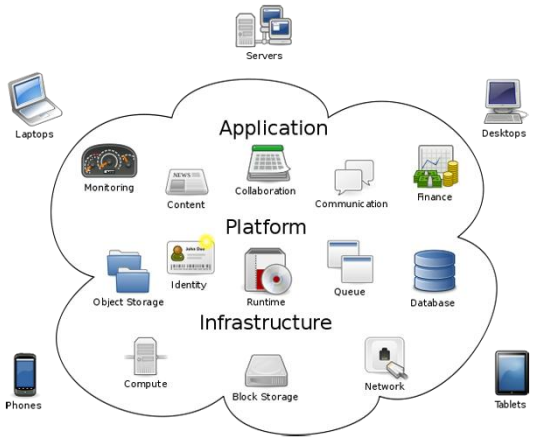


Fig. 1.7 cloud computing service model [41]

1.9.1 Software-as-a-Service

SaaS or “software on demand” gives the client the capability of using software application remotely through the internet using web browser [20]. SaaS is described as a model where an application is hosted as a service to a customer via the internet. The SaaS is also referred to as the cloud application layer, as it is the most visible layer to the end-user [1]. The benefit of SaaS to the client or customer includes; low cost, because the client doesn’t need to employ other mode of traditional software delivery, licensing fees, installation cost, and maintenance fees. The customer only needs to subscribe to the SaaS model of software delivery. Example of applications in the SaaS category includes; the Salesforce’s Customer Relation Management (CRM) system, video conferencing, IT services, accounting etc [20].

1.9.2 Platform-as-a-Service

The PaaS is also known as cloud software environment layer. It provides the client with the ability to develop and publish a customized application in a given environment that is already hosted via the internet. PaaS supplies the client/customer all sort of resources needed to build applications and services completely from the internet without the need to download or to install software. Example of PaaS include Salesforce.com, through their CRM solution, Google’s AppEngine which provides developers with a Python runtime environment, Java, PHP, and other specified APIs to develop applications for Google’s cloud environment [21]. Despite this remarkable achievement, PaaS lacks interoperability and portability among providers [6]. The core cloud interoperability problem is that cloud providers have not done a good job coordinating the use of languages, data, interfaces and other subsystems that are now largely proprietary. Regardless of the interoperability and portability issues, PaaS also exhibits some benefits just like SaaS which includes low cost, providers of the platform handles maintenance and upgrading of the system. This results in efficient and cost effective for enterprise software development.

1.9.3 Infrastructure-as-a-Service

IaaS or Hardware-as-a-Service doesn't provide application to the customer, but rather provide the clients or customers with hardware such that the organization can put whatever it deems to put. IaaS provides virtual machine, virtual storage, virtual infrastructures, and other hardware assets as resources that the clients need [4]. The computational resources which are provided to the clients by virtual machine are the common form of IaaS. The most common example of IaaS includes; the Amazon Elastic Compute Cloud (EC2), Elastic Computing Infrastructure, others include Eucalyptus and Nimbus which are considered as academic source projects [6].

Apart from the computing resources; there is the virtual storage within the cloud model. The cloud storage is referred to as Data-Storage-as-a-Service (DaaS) which allows the user to obtain a flexible storage that can be accessed remotely. DaaS includes Amazon Block Storage (EBS), Simple Storage Services (S3) and Rackspace's, IBM, Blue Cloud, Eucalyptus, FlexiScale, Joyent, cloud files etc [11].

1.10 Client Side

As cloud computing involves the provider and the customer, there should be a medium through which the customer could get to the providers' domain. This can be achieved through the help of client. Client is an interface that consists of hardware, or software that relies on cloud computing for application delivery, or that is specifically designed for delivery of cloud services and that in either case is essentially useless without it [22]. The cloud customers get access to the cloud provider through these two main categories of cloud client, i.e., hardware and Software client. Each of these categories are further subdivided. The figure below (fig. 1.8) shows how these client serve as an interface to the cloud user.

Fig. 1.8 cloud computing clients [22]

1.10.1 Hardware Client:

The hardware client consists of thick client, thin client and smart phones, which are explained below;

Thick client: the thick client involves a device that doesn't need or rely on a server to run [23]. It consist of internal memory, I/O devices and so on. The thick client is a full featured computer which is always functional whether it is connected to a network or not [17]. A personal computer is a typical example of a thick client.

Thin client: These are bare bones computers that allow users to access programs, files and functionality that are hosted on another computer [24]. The cloud client doesn't have any hard drive and no installed software, The thin client runs programs and access data remotely on a given server, Onlive hardware is an example of thin client[17].

Smartphones: are also hardware that allow the cloud customer to access services remotely at any given time; devices like iPhone, Android based phones, and Windows Mobile based phones are examples of the smartphones category.

Customers use the client to access various services in the cloud through web-services. For instance, the cloud services that can be used with thick client

includes; the Amazon Simple Storage Services (S3), the Elastic Compute Cloud (EC2) or Microsoft OneDrive, etc. The OneDrive is a file hosting service that allows users to upload and sync files to cloud storage and then access them from a Web browser or their local device. The thin cloud consists of application like onlive, which provide game-on-demand which runs on the cloud server. The smartphones uses cloud services like the Salesforce.com which is a cloud based CRM system for companies, and Mobile lite client.

1.10.2 Software Client:

The software clients include; rich or fat client, smart client and web-applications/thin client [17].

Rich or Fat client: this includes desktop applications connected to the internet or applications that make use of network support, but run offline. The clients in this group can be email clients, Microsoft Outlook or the media player like iTunes.

Smart Clients: These are interconnected device that allows the user's local application to interact with server-based application through the use of web services [25]. The smart client can work with data even when it is offline. Installation and updating is done automatically through the internet, the applications can run on almost all devices that has internet connectivity.

Web-applications/Thin client: this is an application that runs in a web browser or created in a browser-supported programming language [26]. The web applications rarely need to be installed by the user. Examples of such application include online retail sales, online auction, online agenda etc [17].

1.11 General Overview

The management of resources requires putting a limited access to the pool of shared resources. No matter what kind of resources you are dealing with, it also controls the status of current resource consumption. Resources in Information Communications Technologies (ICT) are the fundamental elements like hardware part of the computer systems, data communications and computer networks,

operating system and software applications. Since the number of these resources is limited, it is important to restrict access to some of them, so as to ensure SLA (Service Level Agreement) between the customers who are requesting resources and providers who are the owners of the systems. Main resource sharing function of a distributed computer system is to assign user requests to the resources in the system such that response time, resource utilization, network throughput are optimized.

As the clients in the cloud ecosystem are increasing, it's good to find an efficient way to handle the clients' demand by maximizing the throughput and minimizing the response time for a given system. The thesis will look in to some literature involved in allocating resources dynamically in the cloud computing ecosystem and then employ some queuing system models with the aim of exploring the most efficient in terms of throughput of the system, time delay and the response time of a given system when dealing with request for resources.

The remaining part of this thesis is organized as follows; chapter two introduces the concept of resource allocation and some strategies in handling resources upon request from a cloud client. Chapter three provide some queuing analysis that were employed in thesis with regard to efficiency in handling a request for resources from the client, by analysing the issue when a single server is used and when a server is virtualized into a pool of multiple servers and also analyse a technique for balancing load across computing resources using max-min algorithm. Chapter four contains some simulation results with regard to the formulated modules in chapter three and lastly conclusions, recommendations and bibliography are stated in chapter five.

Chapter Two: Resource Allocation

2.1 Resource Allocation in Cloud Computing

Resources in cloud computing cover all useful entities which can be use through the cloud platform. These resources include storage, memory, network bandwidth, and virtual machine [7]. The resources can be virtualized and provisioned from the existing physical resources in the cloud environment. The parameters that are virtualized include; the CPU, memory, disk etc. The provisioning can be done by mapping these virtualized resources to their corresponding physical ones. Resource allocation in cloud computing is all about assigning available resources to a needing cloud application. Dynamic resource management is seen as a very active research area in the field of cloud computing. The cloud computing resources costs vary depending upon the type of configuration for using such resources. There for an efficient use of these resources is considered as a prime interest for both the customer/client and the cloud provider. Resource allocation in cloud computing takes place in two levels [28];

- If an application is uploaded to the cloud, a load balancer assigns the requested instances to a physical machine to balance the computational load of multiple applications across physical computers
- If an application receives multiple incoming requests, such requests are assign to a specific application instance to balance the computational load across a set of instance of some applications

Resource allocation exhibits some benefits irrespective of the organization size or business market. It also have some limitations, below are some set of advantages and limitation of resource allocation [42];

Advantages

- Users do not have to install software or hardware to access the applications, develop application and to host the application over the net.

- No limitation of place and medium, application and data can be reached anywhere in the world and on any system.
- Users do not need to purchase the hardware and software systems.
- Cloud providers can share their resources over the internet during scarcity of resources.

Limitations

- Users do not have control over the resources since they rent the resources from a remote server.
- Migration issue occurs when a user decides to switch to other providers.
- In public cloud, security is major issue as clients/customers are concerned that their data could be hacked.
- Peripherals devices like printer and scanner might not work with cloud as many requires software to be install locally.

2.2 Resource Allocation Strategy

Resource allocation strategy involves integrating the cloud provider activities in order to utilize and allocate scarce resources within the limit of cloud environment so that the need for cloud application could be met. For an optimal resource allocation strategy the following issues must be avoided [29].

- Resource contention: when two applications try to access the same resources at the same time.
- Resource scarcity: occurs when resources are limited and demand is high.
- Resource fragmentation: when resources are enough but they are fragmented into small entities and isolated.
- Over-provisioning: when application gets too much of needed resources
- Under-provisioning: When application get less than what its demands.

2.3 Dynamic Resource Allocation

As cloud users increase, resource allocation needs to be made available for the users of the cloud; dynamic resource allocation is seen as the solution to this kind of situation. Static resource allocation in traditional IT infrastructures assigns fixed computing resources to a particular application to satisfy its peak load requirement. However, this results in under utilization of such resources (computing resources), also sometimes the average-load-base static resource allocation scheme assign computing resources to applications based on the average workload of such application. This however can sometimes fail to satisfy the peak load request of the application. Cloud computing offers a solution to such kind of challenge by flexibly managing the resources in dynamic approach [30]. The dynamic resource allocation is achieved through virtualization technology which abstracts, encapsulates and partitioned the computing resources. Today, there exists many dynamic resource allocation approaches, some of which will be analyzed in this thesis. Among all this approaches the first to handle this kind of situation is the dynamic resource allocation for parallel data processing called Nephele.

2.3.1 Dynamic Resource Allocation for Parallel Data Processing

Nephele is the first data processing framework to include a dynamic approach in allocating or de-allocating distinct resources (computing resources) when scheduling and during job execution. In Nephele the user needs to start a VM in the cloud before submitting job. The VM runs a job manager JM which is responsible for scheduling and coordinating the client job received. The JM has the capabilities of communicating with the cloud controller as indicated in the figure below (fig. 2.8). The execution of task is carried out by some set of instances. These instances run what is known as task manager (TM) whose responsibility is to execute a given job and inform the JM for completion or any possible error encountered [29].

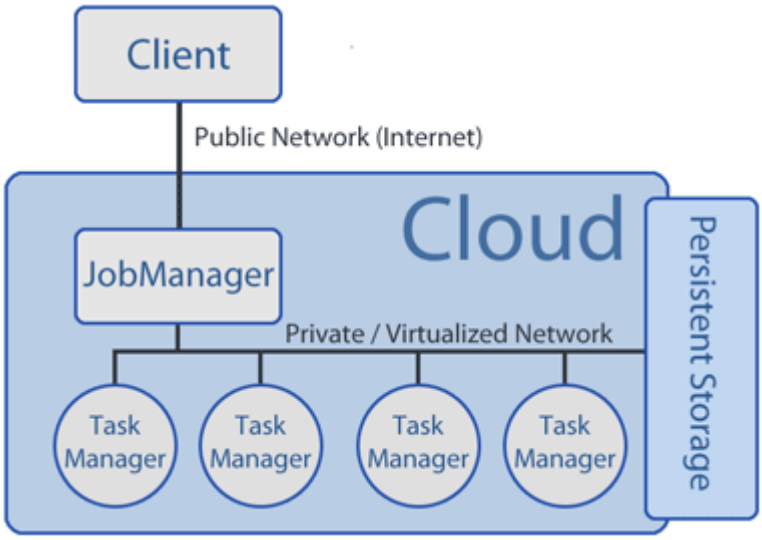


Fig. 2.1 Nephele Architecture. [43]

The persistent storage stores the job's input data and then eventually receives the output data. The persistent storage is accessible to the job manager and the task manager [31]. The Nephele architecture is described in [32] in form of modules as follows;

- Network module: It is a distributed application architecture that partitions tasks or workloads between services provider (i.e. server) and services requester (client).
- Scheduling Task: The client is the one that initiate the task to be processed to the job manager, the JM then read and dispatches the task to a task manager which then allocate the resource for processing.
- Client module: the client who initiates the request to the JM will schedule the process and coordinate the task and then wait for response upon completion.
- Job manager module: The job manager waits for the client to send a task, coordinate and check for the availability of a server. If the server is available the JM allocate the resources for execution and waits for response from the task manager as shown in the flowchart (fig. 2.2) below.

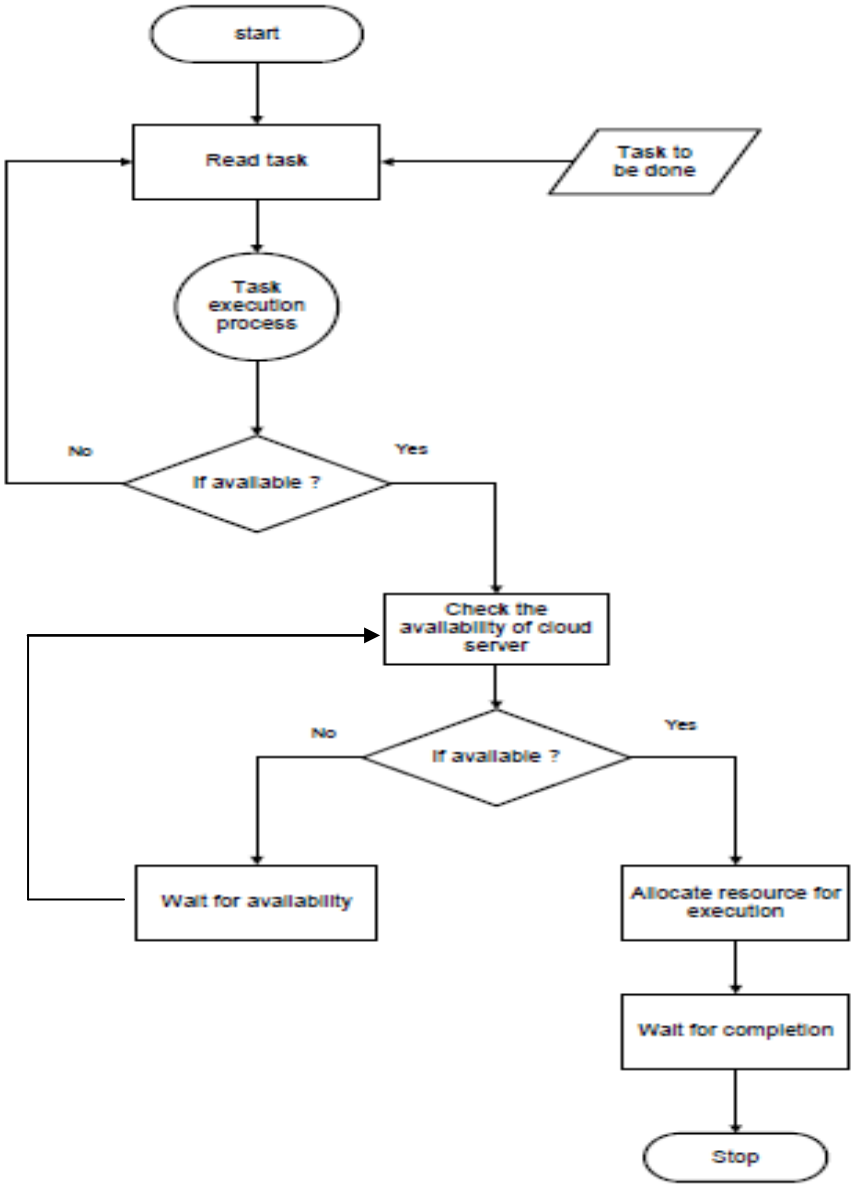


Fig. 2.2 Job Scheduling [32]

- Cloud controller: it is an interface between the job manager and the task manager. It provides control and initiates task manager. The cloud controller coordinates and manages the execution and dispatching of task. It checks for the availability of task manager and allocate resources for execution as shown in the chart (fig. 2.3) below.

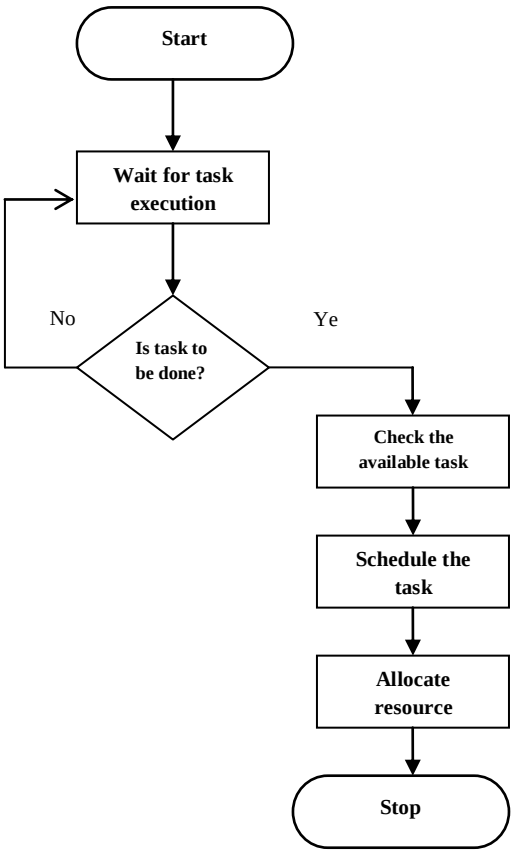


Fig. 2.3 Cloud controller

- Task manager module: the task manager will wait for task to be executed and then inform the job manager which then sends response to the client.

The Nephele's approach for dynamic resource allocation improves the efficiency in scheduling algorithm in real time cloud computing services and it is seen as an optimal approach for resource allocation as it was able to avoid those resource constrains that other approaches experienced. The algorithm works in a manner that early completion tasks are given high priority and less priority for abortion.

2.3.2 Dynamic Resource Allocation using Virtual Machine

Another approach to dynamic resource allocation is the virtual machine approach. Virtualization technology is used to allocate virtual data center resources based on application demands. The goal of using VM approach is to avoid overloading, i.e. the capacity of a physical machine should be sufficient to satisfy the resources needs of all VMs running on it [28]. The techniques involve in this approach include; virtualization technology and skewness.

Skewness: The skewness is the measure of the unevenness in the multi-dimensional resource utilization of a server, by minimizing the skewness of a given server, the overall utilization of the server resources can be improved [52]. This concept is used to compute the unevenness in the utilization of multiple resources on a server [36]. The resources skewness of a server is given as;

$$skewness(p) = \sqrt{\sum_{i=1}^{n} \left(\frac{r_i}{\bar{r}} - 1\right)^2}$$

Where n is the number of utilization of multiple resources on a server p, r_i is the utilization of the i^{th} resource and $\bar{r}$ is the average utilization of all resources for server p.

Virtualization Technology: this technique abstracts the hardware and the system resources from the operating system. The technique is employed in the cloud environment across large set of servers using virtual machine monitor (VMM). The VMM lies between hardware and operating system [36] as shown in the figure (fig 2.4) below;

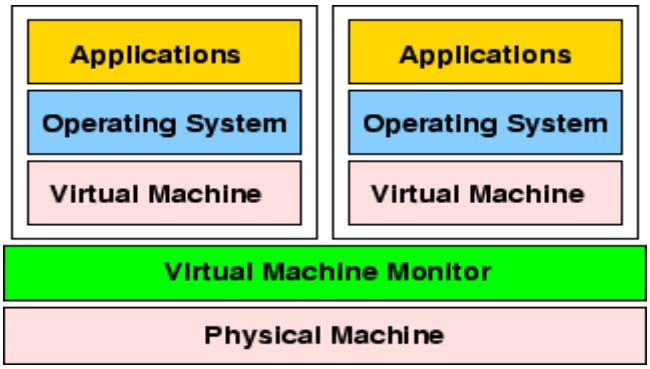


Fig. 2.4 Virtual machine abstraction [40]

Virtual machine monitors (VMM) such as Xen provides a mechanism for mapping virtual machines (VMs) to physical resources. The cloud users have no knowledge of such activities as it is performed on the background on the provider's system.

The virtualization technology helps to handle load balancing dynamically. This makes it possible to remap virtual machines (VMs) and physical resources according to the change in load [45]. Load balancing algorithm allocates efficient VM upon users demand, as it is possible to have multiple requests at a given time. The load balancing algorithm helps the user to decide whether to stay in the queue or look for service by other means. The load balancing technique plays a vital role by distributing large processing load to smaller processing nodes to enhance the overall performance of the system. The figure below depicts a load balancer.

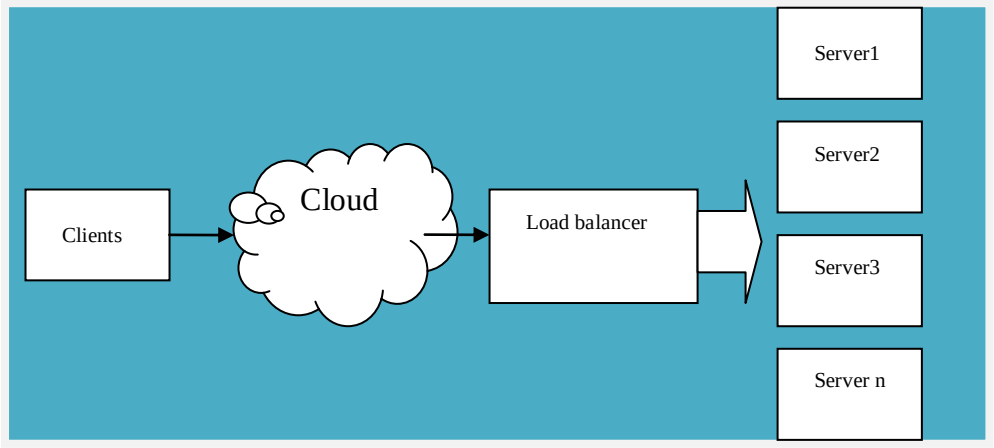


Fig. 2.5: Load Balancing.

2.3.3 Priority Based Dynamic Resource Allocation

The priority based dynamic resource allocation model for cloud computing is another dynamic resource allocation method that is based on virtual machine approach with the aim of minimizing wastage and provide maximum profit. The algorithm makes use of some parameters like; time, cost, number of processors and so on. Whenever a client sends a request the cloud service provider runs the task submitted by the client, look for the task with higher priorities by considering the computational time needed to complete the given task, number of processors needed to execute the task, the importance of the client to the cloud service provider and so on [28]. The scheduling approach employed in the method as stated in [50] is priority based scheduling algorithm to balance the load across various virtual machines.

2.4 Load Balancing:

Load balancing is a technique used to distribute processing load (i.e. large processing load) to smaller processing nodes (i.e. resources) to enhance the overall performance within the system in a distributed environment as shown in fig.2.5 above. The idea of load balancing is to avoid loading up a resource during task scheduling so that all the resources will be allocated with a task evenly across a given virtual environment. Various load balancing algorithms exist as stated in [47] with the aim of distributing the task's load across resources. Some of these algorithms includes;

2.4.1 Round-Robin

This algorithm selects the first node (resource) randomly and then allocate task to all other nodes in a round robin manner until all nodes are allocated. The advantage of round robin algorithm is that it utilizes all the resources in a balanced order. Despite the advantage the round robin algorithm has some setbacks, as some nodes may be heavily loaded while other may not be [46].

2.4.2 Min-Min Algorithm

This algorithm has all the relevant information needed in advance. The algorithm uses some parameters to obtain the information it needs. Some of these parameters are; ETC (Expected Time Compute), MET (Minimum Execution Time), MTC (Minimum Completion Time) etc. The Min-Min algorithm selects a task with minimum completion time and maps it with a node with a minimum completion time [46].

2.4.3 Max-Min Algorithm:

This algorithm works almost the same way as the Min-Min algorithm except in Max-Min the task with maximum value is selected from the set of execution time of tasks and maps it to a node with minimum completion time. The ready time of the node is updated by adding the execution time of the task [46, 47].

2.4.4 Equally spread Current Execution Algorithm

This algorithm handles tasks with priorities. Loads are distributed at random by checking the size, and then transfer the load to a virtual machine that is lightly loaded or that can handle the task easily and takes less time. According to [46] it is a spread spectrum technique, the load balancer spreads the load of a given task into multiple virtual machines.

2.4.5 Throttled Load Balancer

This is another load balancing technique that is also based on virtual machine. The algorithm works by firstly receiving client request that is seeking for a suitable virtual machine to perform the requested operation. As there may be multiple instances of virtual machines, [50] stated that the load balancer will first look for a group, which can handle the request and allocate the process to the lightly loaded instance of that group.

Other load balancing algorithms include; honey bee foraging algorithm, biased random sampling, etc.

Chapter Three: Resource Scheduling

3.1 Resource Sharing Model using Queuing System

This thesis examine the effect in handling resources in the cloud computing environment by employing the queuing concept of single and multiple server systems. In this chapter we will look into the mathematical issues involved in single and multiple server systems. Though cloud computing companies employ the concept of multiple server system and each server is virtualized into another pool of multiple servers, it is important to compare the performance of the two concepts as each server in a multiple server system can also stand as a single server system. The main issue is to increase the performance of the system by reducing overloads so that the system will be able to handle and allocate requested resources effectively.

In cloud computing environment the cloud providers are concerned with how to generate revenue by maximizing the throughput they can serve their clients and to minimize delays in allocating resources to the clients. This will help also in lowering the level of occupancy or utilization rate by the client as the client is more concerned with efficiency of the system.

The issue of load balancing among various VMs is very important in the cloud ecosystem, because it makes it possible to distribute tasks among different resources for execution. The thesis focus on a particular algorithm called the Max-Min algorithm balancing the task accordingly on various resources to enhance the overall performance of the cloud system.

3.2 Queuing Theory

The queuing theory is the study of waiting line. It enables mathematical analysis of related processed which include; arrival, waiting in the queue and been served by a server [12, 17, 34]. Typically queuing model represents first, the system's physical configuration by specifying the number and arrangement of the servers and secondly, the stochastic nature of the demands by specifying the

variability of the arrival process and in the service process. Queuing process exhibit some characteristics which are;

- Arrival pattern of request: the arrival process is usually stochastic, therefore its probability distribution can be determined.
- Service pattern of request: probability distribution is also needed here to describe the sequence of customers' service time.
- Queue discipline: queue discipline refers to the manner in which the requests are selected for service. The discipline includes FCFS (first come first serve) RSS (random service selection) and priority systems where a particular customer is given priority upon arrival to the system.
- System capacity: the queue can be finite or infinite. If there is restriction on when to enter a system then it's called finite a queue system, and if there is no restriction it is called an infinite queuing system.
- Number of service channels: a queuing system can be single or multi-server system.
- Number of service stages: a queuing system may have only a single service stage.

3.2.1 Kendall's Notation

To understand and use the queuing system successfully, Kendall's suggestions describe and classify the queuing system. A typical Kendall notation is given as [29, 33, 34]:

- A: arrival time for request
- S: service time
- C: number of servers

Other three notations include number of buffers, i.e. available place in the system (K), population size i.e. restrictions on a server (N) and service discipline (SD). These notations are arranged in the order as given below;

{Arrival time}/ {service time}/ {number of servers}/ {buffer}/ {queuing discipline}

The arrival and service time follows a specific pattern which can be in one of the following;

- Exponential M
- Deterministic D
- Erlange type EK
- Mixture of k exponential HK
- Phase type PH
- General G

The thesis will make use of the Markovian process whereby the arrival rate follows Poisson process or exponential for the service time. If an even follows Poisson process then its mean is λ, and for the service time it follows the exponential distribution with mean value given as 1/μ [10]. The two Markovian process involve in the thesis include the single server system, m/m/1 and the multiple servers system, m/m/k as both of the techniques can be applied in the cloud computing environments when dealing with sharing of resources.

An important issue that needs clarification is the occupation rate or server utilization (ρ) for both single server (M/M/1) and multiple server (M/M/k). The occupation rate in a single server system is given as the arrival rate multiplied by the service rate [12].

$$\rho = \lambda/\mu \quad (1)$$

To avoid the growing of the queue to infinity the server utilization is required to be less than one, i.e. $\frac{\lambda}{\mu} < 1$. For the multiple server systems it is required that the server utilization should be;

$$r = \lambda/(\mu k) \quad (2)$$

Where k is the number of servers in the system

3.2.2 Single Server System (M/M/1)

In single server queuing model, the system consists of; exponentially distributed inter-arrival time, exponentially distributed service time, one server, infinite number of buffers, infinite population size and first come first serve service discipline. The fig. 3.1 below shows a typical single server queuing model.

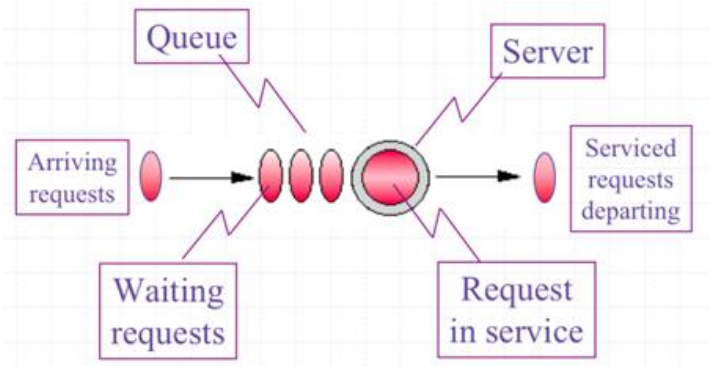
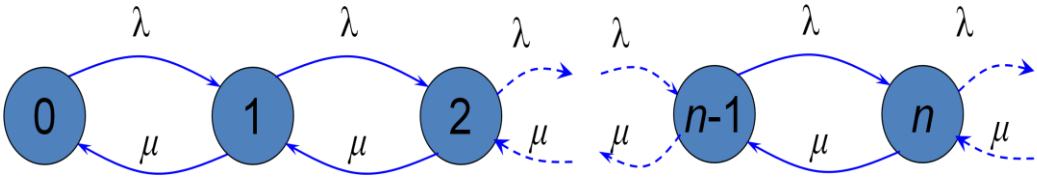


Fig. 3.1 a single server queuing model

The M/M/1 system is build using the concept of birth-death chain, i.e. all the birth rate $\lambda_i=\lambda$ and all the death rate $\mu_i=\mu$ as shown in the fig. 3.2 below;



$$P_n = \left(\frac{\lambda}{\mu}\right)^n P_0$$

for n= 1, 2, 3…

By using normalization condition and the fact that the $\frac{\lambda}{\mu} < 1$ the geometric sum will be

$$P_0 \left[1 + \sum_{n=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^n\right] = 1$$

$$P_0 = 1 - \lambda/\mu = 1 - \rho$$

$$\therefore P_n = (1-\rho)\rho^n \quad (3)$$

This gives us the probability of n request in the system

Performance Metric in single server system

Below are the performance metrics that are used in this thesis;

1: The expected queue length or number of request in the system is given as

$$R = \sum_{n=0}^{\infty} nP_n = (1-\rho)\rho \sum_{n=0}^{\infty} n\,\rho^{n-1}$$

Taking the ant-derivative of the above equation and then rearranging the equation as

$$(1-\rho)\rho \sum_{n=1}^{\infty} n\,\rho^{n-1} = (1-\rho)\rho \sum_{n=0}^{\infty} \frac{d\rho^n}{d\rho}$$

$$(1-\rho)\rho \frac{d}{dn}\sum_{n=0}^{\infty} \rho^n = (1-\rho)\rho \frac{d}{dn}\left(\frac{1}{1-\rho}\right)$$

$$\therefore \mathrm{R} = \rho/(1-\rho) \quad (4)$$

2: The mean response time of the system is obtained using Little's law as

$$T_r = R/\lambda = 1/(\mu - \lambda) \quad (5)$$

3: Expected number of request in the queue is given as

$$Q = \sum_{n=0}^{\infty}(n-1)P_n = \sum_{n=0}^{\infty} n\,p_n - \sum_{n=0}^{\infty} p_n$$
$$Q = R - \rho = \rho\text{^}2/(1-\rho) \quad (6)$$

4: The expected time a request spent on the queue waiting (delay) to be served can be obtained using (5) as;

$$T_q = \mathrm{Q}/\lambda = \rho/(\mu(1-\rho)) \quad (7)$$

5: The throughput for the system can be obtained as

$$Th = \mu \sum_{n=1}^{\infty} p_n = \mu(1-p_0) = \mu\rho$$

$$Th = \lambda \quad (8)$$

3.2.3 Multiple Server System

In multiple server system, the request pattern, like that of single server system, assumes Poisson arrivals, exponentially distributed service times, identical servers and infinitive capacity buffers. The request that arrived in the buffer will be served by a single server in the system that is idle. The servers are identical and any request can be served by any server as shown in fig. 3.2 below;

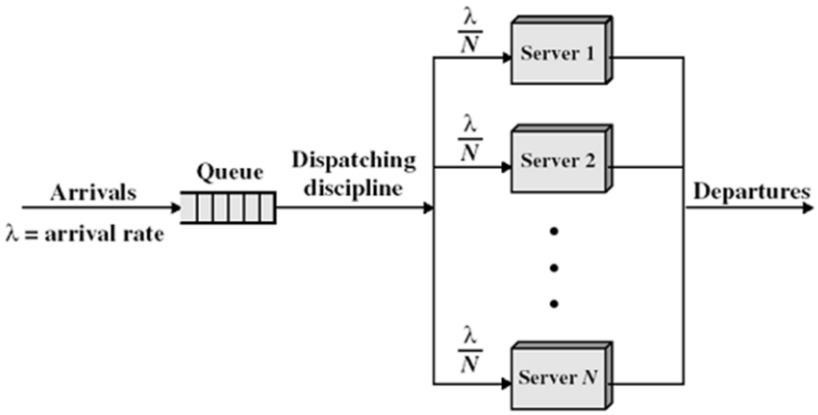
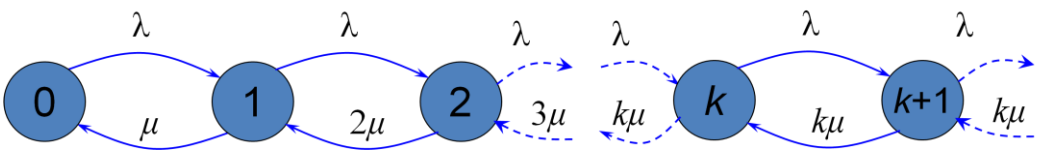


Fig. 3.2 a single server queuing model

By employing the general birth-death results again with relationship below. The performance metric of the multiple server system can also be obtained.



$$\lambda_n = \lambda \, and \, \mu_n = \begin{cases} n\mu \; if \; 0 \leq n \leq k \\ k\mu \;\; if \; n \geq k \end{cases}$$

Now the general birth-death result is given by

$$P_n = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n p_0 if \; n < k, \text{ and } p_n = \frac{k^k}{k!}\left(\frac{\lambda}{\mu}\right)^n p_0 \; if \; n \geq k$$

For $\rho = \frac{\lambda}{k\mu}$ then the above equation can be as below

$$P_n = \begin{cases} \frac{(k\rho)^n}{n!} P_0 \; if \; n < k \\ \frac{k^k \rho^n}{k!} P_0 \; if \; n \geq k \end{cases}$$

P_0 is obtained the same way as in the case of single server system M/M/1, as follows;

$$P_0 \left[1 + \sum_{n=1}^{k-1} \frac{(k\rho)^n}{n!} + + \sum_{n=k}^{\infty} \frac{k^k \rho^n}{k!} \right] = 1$$

$$P_0 = \left[1 + \sum_{n=1}^{k-1} \frac{(k\rho)^n}{n!} + \sum_{n=k}^{\infty} \frac{k^k \rho^n}{k!} \right] = 1$$

$$p_o = 1 \Big/ \left\{ \sum_{n=0}^{k-1} \frac{(k\rho)^n}{n!} + \frac{(k\rho)^k}{k!\,(1-\rho)} \right\}$$

However the utilization rate for a single server is given by r= $\frac{\lambda}{k\mu}$ therefore the above equation can be rewritten as

$$p_o = 1/\{\textstyle\sum_{n=0}^{k-1} r\text{^}n/n! + r\text{^}k/(k!\,(1-\rho))\} \qquad (9)$$

Performance Metric in M/M/k Server System

1: The expected number of requests in the queue or mean request is given by

$$Q = \sum_{n=k}^{\infty}(n-k)P_n = \sum_{d=0}^{\infty} d\, P_{n+d} = \sum_{d=0}^{\infty} \frac{d\left(\frac{\lambda}{\mu}\right)^{k+d}}{k!n^d} P_0$$

Like in M/M/1 the anti-derivative concept is used and then the summation of ρ^d is found as given in the equation

$$= \sum_{d=0}^{\infty} d\, \frac{\left(\frac{\lambda}{\mu}\right)^k}{k!}\rho^d P_0 = P_0 \frac{\left(\frac{\lambda}{\mu}\right)^k}{k!}\rho \sum_{d=0}^{\infty} \frac{d\rho^d}{d\,\rho} = P_0 \frac{\left(\frac{\lambda}{\mu}\right)^k}{k!}\rho \frac{d}{d\,\rho}\sum_{d=0}^{\infty}\rho^d$$

$$\therefore Q = P_0(((r)\text{^}k\,\rho)/(k!\,(1-\rho)\text{^}2\,)) \quad (10)$$

2: Another important issue is the expected time (delay), a request spends in the queue, which is obtained by dividing (11) by λ.

$$T_q = P_0 \frac{(r)^k \rho}{k!(1-\rho)^2} \div \lambda$$

$$\therefore T_q = P_0((r)\text{^}k/(k!\,(\mu k)\,(1-\rho)\text{^}2\,)) \quad (11)$$

3: The mean request in the service facility is obtained as

$$n = \sum_{n=k}^{n-1} kP_k + \sum_{k=n}^{\infty} nP_k$$
$$= r\left(\sum_{n=k}^{n-1}\frac{r^k}{k!} + \frac{r^n}{n!(1-\rho)}\right)P_0$$
$$n = r\frac{1}{P_0}P_0 = r \quad (12)$$

4: Now total expected number of request in the system is given as

$$R = Q + n$$
$$= (P_0((r)\text{^}k\,\rho)/(k!\ \ 〚(\mu k)(1-\rho)〛\text{^}2\,) + r) \quad (13)$$

5: So the mean response time can be obtained using (13) as

$$T_r = R/\lambda \quad (14)$$

6: In a case where a request arrives at the system and finds that the servers are busy, then is forced to join the queue. The probability (time) that the servers are busy is obtained using Erlang C formula.

$$P_{busy} = \sum_{n=k}^{\infty} P_n = r\hat{}k/(k!\,(1-\rho))P_0 \quad (15)$$

7: Another important issue employed in this thesis is the throughputs for the multiple server system, the throughput of a completed service in a given time is obtain as

$$Th = k\rho\mu$$

$$Th = k\lambda \quad (16)$$

These performance metrics will be used in the thesis to see how efficient resource sharing is when a different queue model is employed. However it is obvious that a multiple server system will be more efficient in terms of performance. However, it is important to put these facts into analysis, as it will help researchers to easily visualize the differences when employing such systems. Even in multiple server systems, those with more servers will perform better that those with less servers. Virtualization of these servers will make the systems more efficient when allocating resources to different user request.

3.3 Task Scheduling in Cloud Computing

Task scheduling is a well known concept as it is a vital aspect in cloud computing. It allows for scheduling virtual resources over the cloud to keep a balance load across the resources. The figure below (Fig. 3.3) shows how scheduling is performed across the cloud environment.

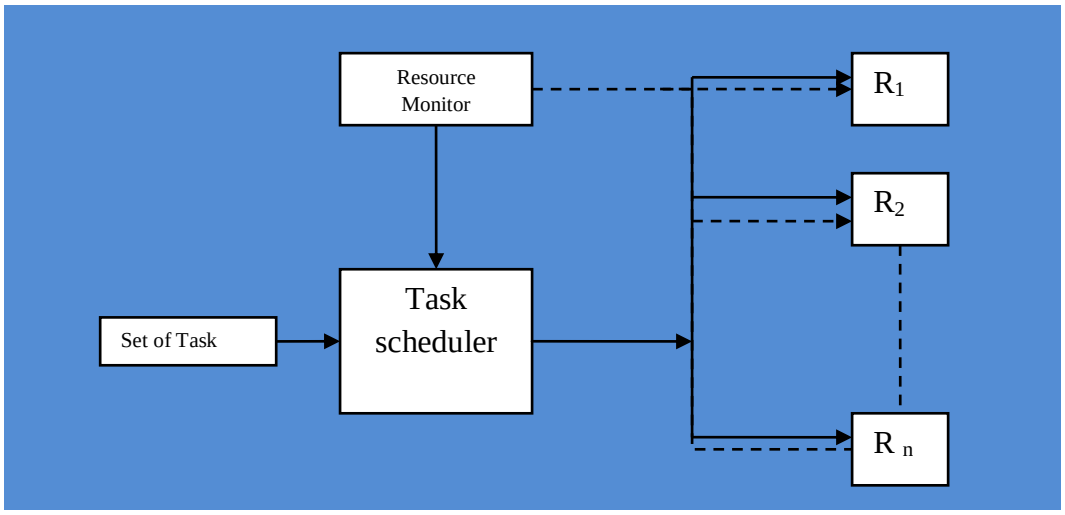


Fig. 3.3 Task Scheduler for Virtual Resources

As it is known, the users send task to the cloud environment with different requirement to the cloud service providers. The requirement can be tasks with different set of data size and processing power, the task scheduler will then match the tasks with available resources (virtual resources) that are available. Some mathematical relations are given in [48] to analyze resources scheduling in cloud computing which are employed and used in this thesis are given below.

The set of VMs V with their respective processing power is given as;

$$V = \{v_j(c_j) | \, j = 1,2, \dots w\} \tag{17}$$

The set of tasks is also given as

$$T = \{t_i(s_j, q_j) | \, i = 1,2, \dots y\} \tag{18}$$

Where

c_j = processing speed (MIPS)

t_i = given task i

s_i =Data file size of a given task (Mb)

q_i = Processing power (MI) of a task t_i

With above equations (17) and (18) the expected execution time (EET) for a given task by a virtual resource can be obtained as;

$$EET = (Size\ of\ task\ (MI))/(Computing\ Power\ of\ resource\ (MIPS)) \quad (19)$$

Now with equation (19) above, another metric can be obtained, which is the completion time (CT) of task t_i by a given resource.

$$CT_{(i,j)} = EET_{(i,j)} + r_i \quad (20)$$

Where r_i indicate the starting time of the execution of task t_i.

Using (20), another important metric can be obtained, which is called the makespan, define as a measure of the throughput of the heterogeneous computing system [47].

$$makespan = Max_{i \in w, j \in y)}(CT_{i,j}) \quad (21)$$

This thesis employs a known scheduling algorithm called an improved max-min algorithm from [49] and then based on this algorithm proposes another algorithm that will help in balancing load across the VMs' resources to improve the performance of the system.

3.3.1 Improved Max-Min Algorithm

The Max-min algorithm allocated task t_i to resource v_j such that large tasks have higher priority. For instance for a given large task, the max-min algorithm execute smaller task concurrently while running large tasks. Therefore, the largest task determines the total makespan for other resources. The improved max-min algorithm is given below [50].

```
for all submitted tasks in Meta-task; t_i
     for all resources; v_i
     C_ij = E_ij + t_j
     Find task t_k costs maximum execution time
     Assign task t_k to resource v_i which gives minimum
completion time
Remove task t_k from Meta-tasks set.
Update t_j for selectedv_j.
Update c_ij for all j.
While Meta-task not Empty
     Find task t_k costs maximum execution time.
     Assign task t_k to resource v_j which gives minimum
completion time
Remove Task t_k form Meta-tasks set.
Update t_j for selectedv_j.
Update c_ij for all j.
```

The algorithm computes the completion time of task submitted to each resource, the task with the highest or largest expected execution time is then assigned to a resource with minimum completion time, and then the task is removed from the meta-tasks set. The meta-tasks set are updated and the max-min algorithm continues until all tasks are assigned to a resource.

3.3.2 Proposed Algorithm

The improved max-min algorithm is reliable and proved to be efficient in scheduling the set of tasks to the available resources. However to make sufficient use of resource a proposed algorithm was introduced which is based on the improved max-min algorithm but small changes are made to make sure that all resources are used sufficiently and to minimize the use of these resources if few once can perform the task. The proposed algorithm is shown in the pseudo code below;

```
For all submitted tasks in Meta-task; Ti
     For all resources; Rj
          Cij = Eij + tj
          Find task Tk costs maximum execution time
          Assign task Tk to its corresponding resources Rj
Remove task Tk from Meta-tasks set.
Update rj for selected Rj.
Update Cij for all j.
Pivot= Tk;
For all updated task in Meta-task; Ti
     For all updated resources; Rj
          Find task Th costs maximum execution time
               Assign task Th to its corresponding
          resource Rj
Remove task Th from Meta-tasks set.
Update rj for selected Rj.
Update Cij for all j.
2pivot= th
While Meta-task not Empty
     Find task Tg costs maximum execution time.
          If 2pivot+tg ≤Pivot then
               Assign task Tg to previous resource Rj
               which gives minimum completion time
     Remove Task Tg form Meta-tasks set.
     Update rj for Selected Rj.
     Update Cij for all j.
     Updata 2pivot.
     Else
          Assign task Tg to resource its corresponding
     resource Rj
     Remove Task Tk form Meta-tasks set.
     Update rj for Selected Rj.
     Update Cij for all j.
```

In the algorithm the total makespan is made to be a pivot (1) value for the first step and another pivot (2) value is assigned during the second step of the execution. Then during the next execution step the second pivot value and the completion time of the current state are summed up together. If they are greater than the first pivot value, then a new resource is allocated to that task. By given this criteria, the resources can be used in a balanced manner and fewer resources can be used, the remaining resources will not be involved to minimize the use of such resources. The flowchart for the above pseudo code is given in the figure (fig. 3.4) below.

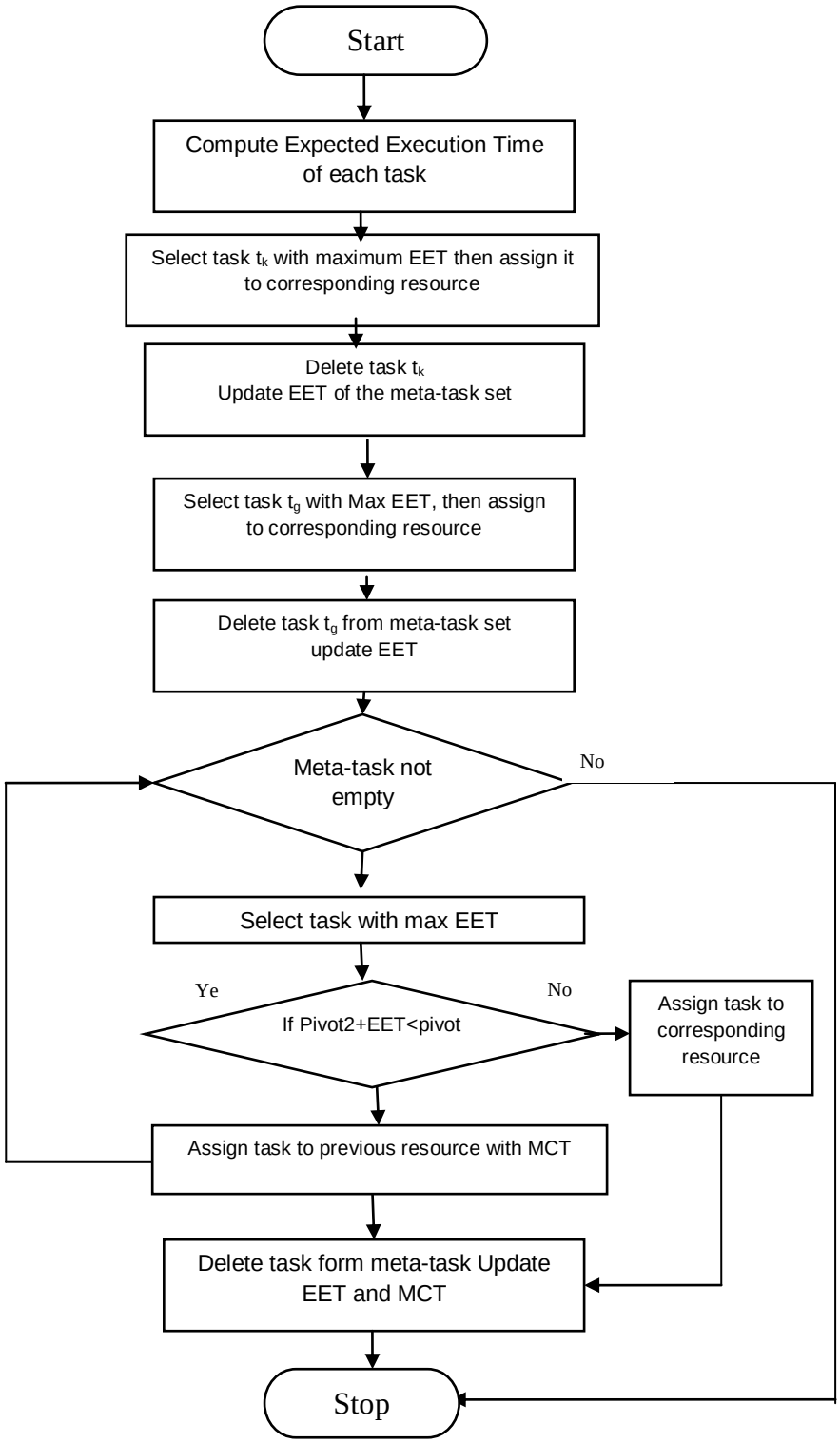


Fig. 3.4: Proposed Algorithm Flowchart

Chapter Four: Analysis and Results

4.1 Queuing Simulation Results and Analysis

This thesis assumes a value ‘k=5’ to be the number of service facilities in the system and ‘n=10’. The performance metrics of the two queuing models are tabulated in a given table and graphics are given to show how each metric affect the system. The arrival rate λ and the service rate μ are given some arbitrary values and taking into consideration the equilibrium status, i.e. $\lambda < \mu$ so as to keep the system within control, i.e., to avoid the continuous growth of the system. The results obtained for both the two systems will be compared numerically to see how each system behaves during resource handling.

As stated earlier, the concept of single and multiple server system is a well known concept, but it is important to have a visual numerical values that will further show how each concept behaves. This concept can also be viewed in such a way that, a single server system to be a single physical machine which can be virtualized into several virtual machines to form a multiple server system i.e. a multiple server is formed from a single server. This will clearly show the benefit of virtualization of a physical machine.

4.1.1 Time Delay:

The table below shows the time in which a request for cloud resources need to wait in the queue before being served.

Λ	μ	Delay(T_q)	
		M/M/1(s)	M/M/c(ms)
15	25	0.0600	0.0268
30	50	0.0300	0.0134
45	78	0.0161	0.00667
60	95	0.02280	0.0112
85	110	0.0309	0.0147

Table 1: Arrival rate, Service rate and Delay

The graph showing the behavior of the time delayed for the two systems is shown below;

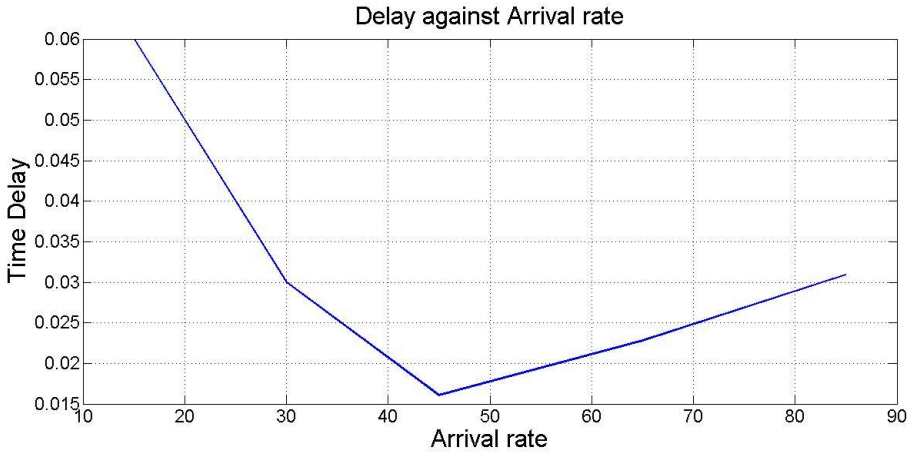


Fig. 4.1: Time delay against the arrival rate for a single server system.

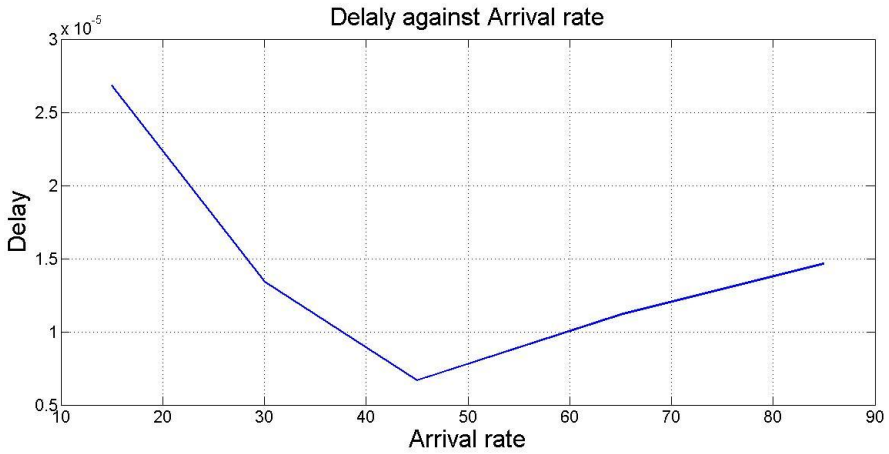


Fig. 4.2: Time delay against the arrival rate for a multiple server system.

The above figures (Fig. 4.1 and fig. 4.2) show how the time a request needs to wait before been served and it clearly show how the delay is minimized in the case of fig. 4.2.

4.1.2 Throughput

The values for the throughput which is seen as the number of completed requests in a given time are given in the table below (table2);

Λ	μ	Throughput(Th)	
		M/M/1	M/M/c
15	25	15	75
30	50	30	150
45	78	45	225
60	95	60	325
85	110	85	425

Table 2: Arrival rate, service rate, and throughput.

The throughput graphs for the two systems are shown in fig. 4.3 and fig. 4.4 below;

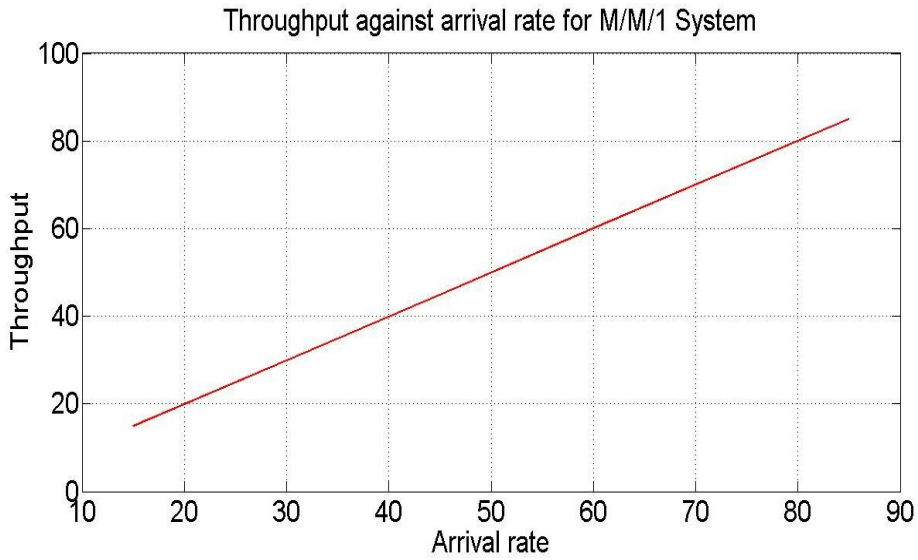


Fig. 4.3: Throughput against the arrival rate for a single server system

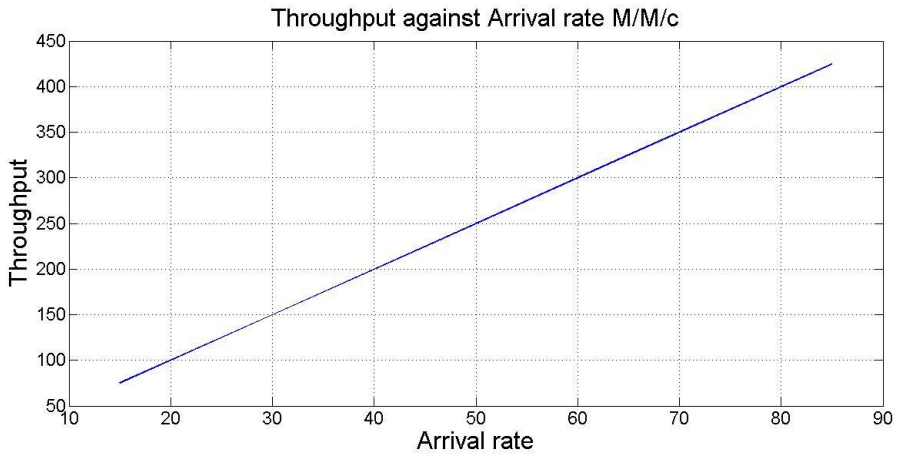


Fig. 4.4: Throughput against the arrival rate for a single server system

From the above figures (Fig. 4.3 and fig. 4.4), we can see that the multiple server system execute more request within a given time than the single server system.

4.1.3 Utilization Rate

The utilization rate or occupancy shows how a system is occupied with requests for resources. The table (table 3) below shows the simulation results for the utilization rate of the two systems;

Λ	μ	M/M/1 (ρ %)	M/M/c (ρ %)
15	25	60	6
30	50	60	6
45	78	57.69231	5.769231
60	95	63.15789	6.315789
85	110	77.27273	7.727273

Table 3: Arrival rate, service rate, and the utilization rate (occupancy)

The utilization rate graph of the tabulated values above for the single and multiple server systems are shown in the figures (fig. 4.5 and fig. 4.6) below;

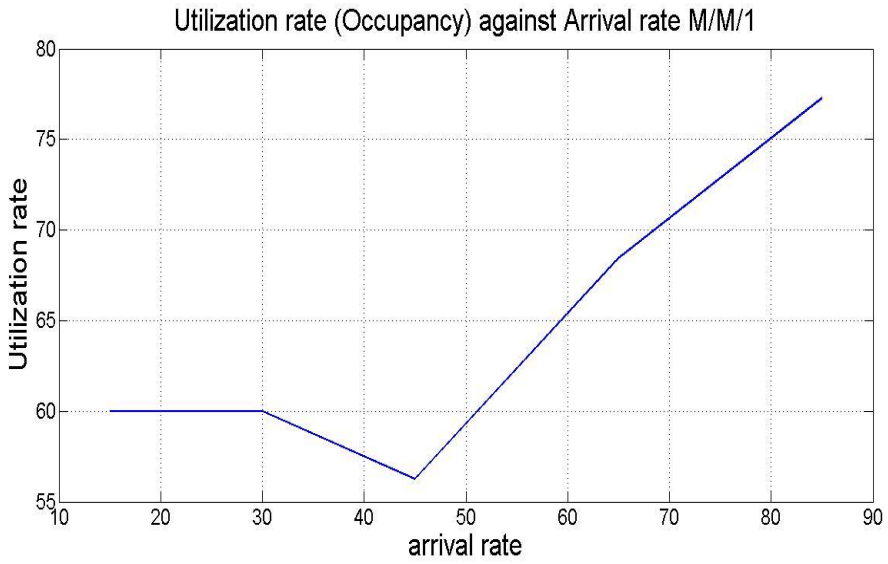


Fig. 4.5 Utilization rate against arrival rate for M/M/1 system.

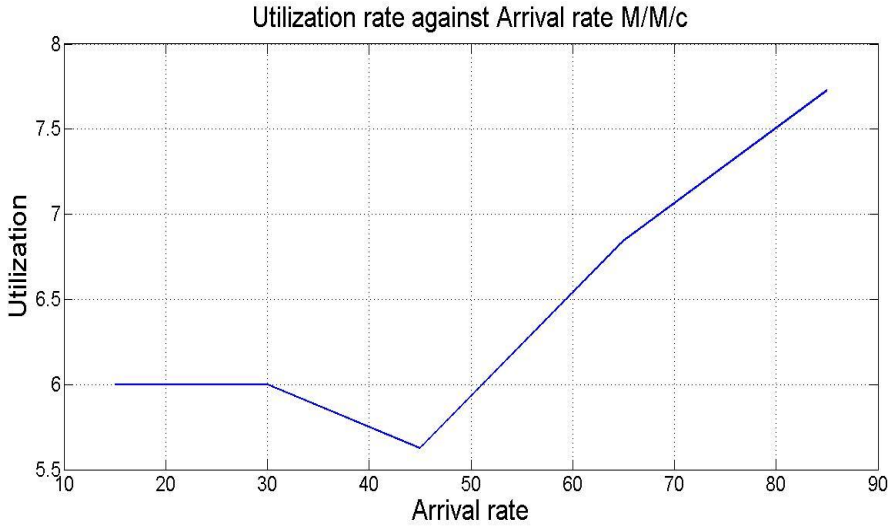


Fig.4.6 Utilization rate against arrival rate for M/M/c system

Fig.4.5 and Fig.4.6 shows the level of utilization (based on the values in Table 2) of a given system when handling the same request. As anticipated the multiple server system is less occupied when compared to the single server system. Hence the multiple server system can handle other requests for resources faster and allocate such resources within a very short time, thereby improving the QoS of the system.

In the case of multiple server system another performance metric used in this thesis is the probability (time) in which a request arrives and then find all servers to be busy and then join the queue. The table below shows the time in which the request joins the queue when all servers are busy.

λ	μ	M/M/c (Pn)(ms)
15	25	0.37834
30	50	0.37834
45	78	0.28333
60	95	0.67675
85	110	1.14907

Table 4: Arrival rate, Service rate, and time request is forced to join the queue

The figure (fig. 4.7) below shows how a request is force to join the queue when all serves are busy, the graph follows the same pattern to the utilization of server, as it is the time a given server may be busy.

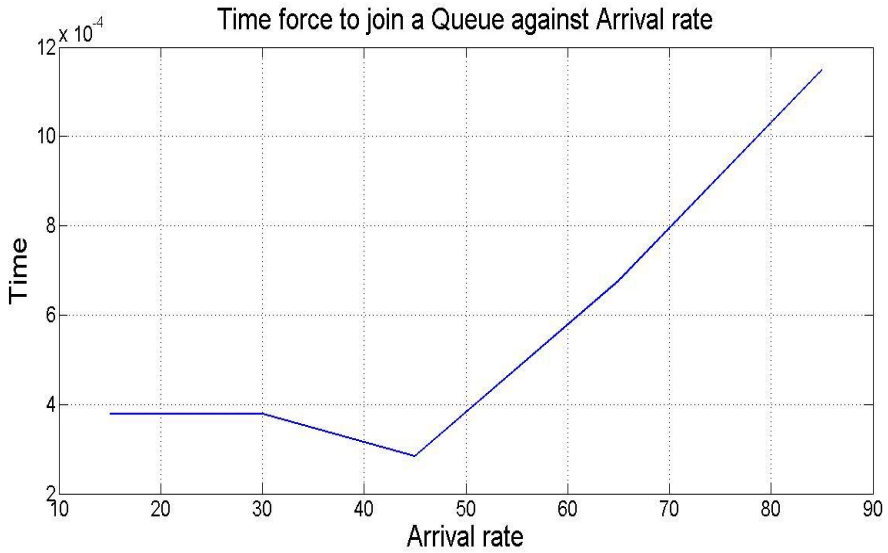


Fig. 4.7 Arrival rate against the Time force to join the queue

4.1.4 Response Time

The response time is the time a request spends waiting on the queue and then received services. Table 4 below shows the value of the response time for both of the two systems employed in this thesis.

Λ	μ	M/M/1 (RT)(s)	M/M/c (RT)(s)
15	25	0.1000	0.0400
30	50	0.0500	0.0200
45	78	0.0286	0.0125
60	95	0.0333	0.0105
85	110	0.0400	0.0091

Table 5: Arrival rate, service rate, and the response time

The graphs for table 4 above are shown below in fig.4.7 and fig. 4.8

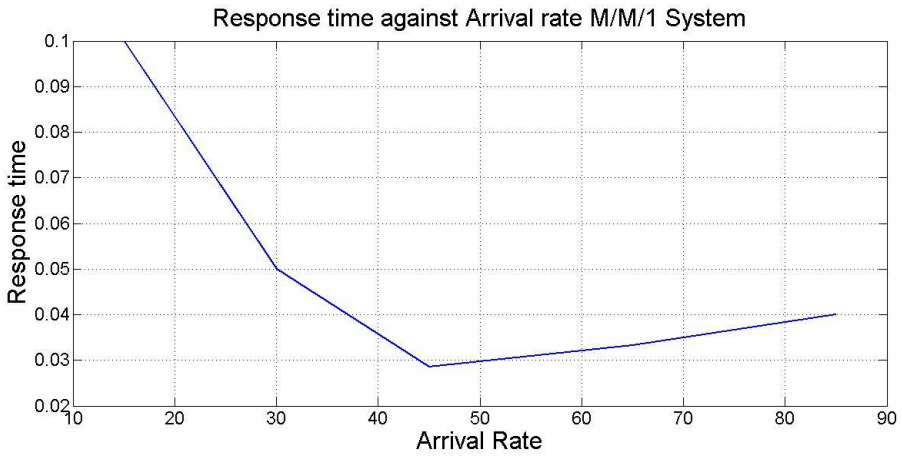


Fig. 4.8 Response time against arrival rate for M/M/1 system.

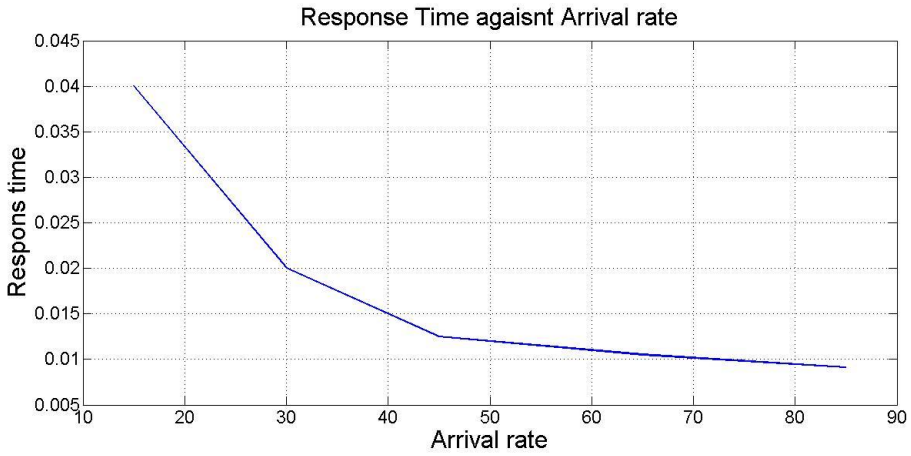


Fig. 4.9 Response time against arrival rate for M/M/c system

From the two graphs above, fig 4.8 i.e., system with multiple server has lower response time when compared with fig. 4.7 i.e. system with single server. Therefore M/M/c system has higher efficiency in terms of allocation of resources to requested clients.

4.2 Max-Min Scheduling Algorithm Analysis and Results

Scenario: Below is a theoretical analysis of some predefined meta-task values and resources used to carry out the scheduling process. The tables below shows the meta-task values and the resources used.

Task	Size of task (MI)	Data volume (Mb)
T1	512	200
T2	1028	500
T3	420	300
T4	330	410
T5	550	328

Table 6: Tasks values

The table below holds the processing speed and the bandwidth of the resources on a network system.

R	Processing speed (MIPS)	Bandwidth (MbPS)
R1	128	100
R2	1256	120
R3	284	150

Table 7: Resource processing speed and bandwidth

Given the above values, Matlab is employed to compute the expected execution time of each task and the results are tabulated and analyzed as given below;

	R1	R2	R3
T1	4	2	1.802
T2	8.031	4.015	3.619
T3	3.281	1.640	1.478
T4	2.578	1.289	1.161
T5	4.296	2.148	1.936

Table 8: Expected execution time of task

From the above tables i.e. table 8: T_i with maximum execution time is selected and then is assigned to the corresponding resource R_i. The Gantt chart below shows how the allocation is performed.

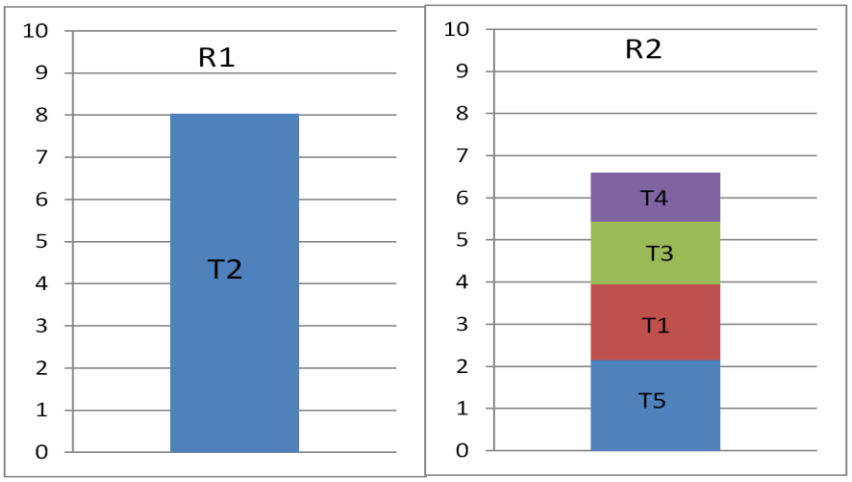


Fig. 4.11: Chart of Resource Allocation for proposed Max-Min Algorithm

From the chart (fig. 4.11), the largest task has a maximum makespan of 8.031 and it's scheduled to resource R1. The maximum makespan, is considered as the maximum throughput for other resources. This makes it possible to balance different smaller tasks to run concurrently on different resources across the system and also to use the resources wisely when needed. Another important factor which is based on the on-demand characteristics of cloud computing is that, the number of resources used is also minimized and a resource can be put into use when there is a demand for that resource. Based on the results obtained, instead of assigning the load to the three resources, it's possible to assign the task to only two resources, thereby increasing the efficiency of the system.

Chapter Five

5.2 Conclusion

In conclusion, cloud computing has become a technology that provides the user with a remote use of pool of resources that are shared over the cloud by the help of other existing technologies like web services, virtualization and others. Researcher proposed different approaches that will enhance the system for an effective and efficient allocation of resources to different cloud clients, with the aim of providing the customers with a better QoS and on the other hand generate more revenue to the cloud computing companies.

In this thesis it is clearly seen that the two queuing models were employed with the aim of improving the efficiency of the system when dealing with resource sharing by investigating the time delay (i.e. waiting time), the throughput of the system. It was found that the multiple server system have lower delay and high throughput that the single server system. The response time in which a client receives service for multiple server system is also lower than the single server system. With the values obtain form the simulation result of both two systems, it is enough to say that the multiple server system is much more efficient and reliable when handling request for resources in the cloud computing environment.

Cloud computing is an on-demand service therefore, efficient on-demand allocation of VM is needed. In the thesis technique to handle on-demand allocation is analyzed. Allocation of resources can be performed efficiently within a cloud environment by balancing the load across the various virtual machine resources, by employing an efficient technique for load balancing such as the max-min algorithm that was used in this thesis. The usage of max-min technique made it possible to handle resources in an efficient and balanced manner. Thus, for a better service to be experienced in a field of cloud computing, a proper and efficient allocation techniques need to be adopted.

5.3 Recommendations

Having seen the behaviors of the two system by investigation, it is recommended that cloud computing systems need to be virtualized into a multiple server system, i.e., to employ a multiple physical machine and then virtualizes each physical machine in to a pool of virtualized machines. It is also important to keep in mind the green computing by minimizing the physical machines and maximizing the virtual machines, which can lead to efficient processing and utilization of computing infrastructure, keep the environment safe, and minimization of energy sustained for the future growth of cloud computing. As cloud computing continues to grow, cloud user (customer) may decide to change/move to other cloud providers, therefore it is recommended that a medium for which client can be able to move their entire data easily and safely from one provider to the other need to be provided to allow mobility among different cloud providers. Also to enhance efficient and reliable QoS, the cloud providers need to adopt a balancing technique that can handle resource allocation effectively and based on the on-demand nature of the cloud computing field.

5.4 References

[1] M Böhm, S Christoph riedl, H krcmar, Cloud Computing and Computing Evolution, *Technische Universität München (TUM), Germany*.

[2] J. Kris, Cloud Computing: SaaS, PaaS, IaaS, Virtualization, Business Models, Mobile, Security and More, 2012 edition.

[3] Introduction to Cloud Computing, White paper. Retrieved from http://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf on 2/4/2014.

[4] Barrie Sosinsky, Cloud Computing Bible, Wiley Publishing. Inc 2011.

[5] Ryan Knight, The new role of XML in cloud data integration Using XML to integrate Salesforce data with enterprise applications. June 2009.

[6] Cloud computing basics. Retrieved from http://south.cattelecom.com/rtso/Technologies/CloudComputing/0071626948_chap01.pdf on 20/3/2014.

[7] Y Yuan, W-Cai Liu. Efficient resource management for cloud computing 2011.

[8] H. Jin, X. Ling, S. Ibrahim, W. Z. Cao, S. Wu, and G. Antoniu, *Flubber: Two-level disk scheduling in virtualized environment*, Future Generation Computer Systems-the International Journal of Grid Computing and E-science, vol. 29, pp. 2222-2238, Oct 2013.

[9] B. Mondal, K. Dasgupta, and P. Dutta, *Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach*, 2nd International Conference on Computer, Communication, Control and Information Technology (C3it-2012), vol. 4, pp. 783-789, 2012.

[10] B. Sosinsky, *Cloud Computing Bible*, Wiley Publishing. Inc 2011.

[11] Fei TENG. Resource Allocation and Scheduling Models for Cloud computing. Jan 2012.

[12] K. A. Williams, *Queuing note, Department of computer science*, North Carolina A & T State University, 2012.

[13] R. Buyya, J Broberg, A. Goscinski. CLOUD COMPUTING Principles and Paradigms. 2011

[14] B. Furht. A. Escalante. Handbook of Cloud Computing. 2010

[15] R. Buyya, R N. Calheiros, X Li. Autonomic Cloud Computing: Open Challenges and Architectural Elements

[16] M Höfer, G Howanitz. The Client Side of Cloud Computing. July 2009

[17] N. T. Thomopoulos, *Fundamentals of Queuing Systems*, Stuart School of Business Illinois Institute of Technology Chicago, IL 60661 USA. 2012 Ed.

[18] Retrieved from https://www.ibm.com/developerworks/community/blogs/sreek/entry/cloud_4?lang=en on 10/4/2014.

[19] Retrieved from http://apprenda.com/library/cloud/introduction-to-cloud-computing/ on 10/4/2014.

[20] Retrieved from https://cloud.google.com/products/app-engine/ on 3/04/2014

[21] Tom Nolle. Retrived from http://searchvirtualdesktop.techtarget.com/contributor/Tom-Nolle on 3/04/2014.

[22] Retrieved from http://www.frustrationfreeit.com/cloud-computing/cloud-clients.html on 18/03/2014.

[23] Retrieved from http://www.tricerat.com/resources/topics-library/thin-clients-and-cloud-computing on 20/03/2014.

[24] Retrieved from http://www.webopedia.com/TERM/S/smart_client.html on 20/03/2014.

[25] Retrieved from http://en.wikipedia.org/wiki/Web_application on 3/03/2014

[26] *James Bond*. Retrieved from http://mycloudblog7.wordpress.com/2013/04/19/the-evolution-to-cloud-computing-how-did-we-get-here/ on 29/03/2014.

[27] N.Krishnaveni G.Sivakumar, Survey on Dynamic Resource Allocation Strategy in Cloud Computing Environment. 2013

[28] R Shelke, R Rajani. Dynamic resource allocation in Cloud Computing. 2013

[29] W Lin, J. Wang, C Liang, D Qi. A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing. 2011

[30] V Kumar, S. Palaniswami. A Dynamic Resource Allocation Method for Parallel Data Processing in Cloud Computing. 2012

[31] Dynamic Resource Allocation for Efficient Parallel Data Processing using RMI Protocol 2013

[32] M Tapkire, B Patil and V Chandode. Parallel Data Processing in Cloud using Nephele, 2013

[33] S. Mohanty, P. K. Pattnaik and G. B. Mund, A Comparative Approach to Reduce the Waiting Time Using Queuing Theory in Cloud Computing Environment, 2014.

[34] Zukerman, Introduction to Queuing Theory and Stochastic Teletraffic Models, 2014, p.88.

[35] T. Sai Sowjanya, D.Praveen, K.Satish, A.Rahiman. The Queuing Theory in Cloud Computing to Reduce the Waiting Time, April 2011.

[36] Satyanarayana, P. Suresh Varma, M.V.Rama Sundari, P Sarada Varma, Performance Analysis of Cloud Computing under Non Homogeneous Conditions, May 2013

[37] M. Bharathi, P. Sandeep Kumar, G. V. Poornima, Performance factors of cloud computing data centers using M/G/m/m+r queuing systems, Sept 2012.

[38] Ivo Adan and Jacques Resing. Queuing Theory. Department of Mathematics and Computing Science, Eindhoven University of Technology. Feb 2001.

[39] János Sztrik. Basic Queuing Theory. University of Debrecen, Faculty of Informatics. 2012

[40] Retrieved from https://www.usenix.org/legacy/event/usenix01/sugerman/sugerman_html/node1.html on 12/05/2014.

[41] Retrieved from http://cloudcomputinginindia.wordpress.com/[18/05/2014]

[42] Retrieved from http://www.frustrationfreeit.com/cloud-computing/cloud-clients.html on 20/04/2014.

[43] Retrived from http://stratosphere.eu/docs/pre-0.4/internals/nephele.html on 19/04/2014.

[44] Retrieved from http://mqitcorporation.wordpress.com/tag/cloud-computing/ on 12/05/2014

[45] M. Rajeswari, M. Savuri Raja, I Thamizheselvan. Resource and Power Management in cloud. International Journal of Scientific Research and Education. Vol. 2 page 460-468. 2014.

[46] D. Manan Shah, A. Amit Kariyani, L. Dipak Agrawal. Allocation of Virtual Machines in Cloud Computing using Load Balancing Algorithm. International Journal of Computer Science and Information Technology & Security. Vol. 3 2013.

[47] S. Swaroop Moharana, D. Rajadeepan. Analysis of Load Balancer in Cloud Computing. International Journal of Computer Science and Engineering Vol.2 2013.

[48] Yichao Yang, Yanbo Zhou. Heuristic Scheduling Algorithms for Allocation of Virtualized Network and Computing Resources. Journal of Software Engineering and Application 2013.

[49] Upendra Bhoi, Purvi N. Ramanuj. Enhance Max-Min Task Scheduling Algorithm in Cloud Computing. International Journal of Application or Innovation Engineering & Management. 2013.

[50] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud. Improved Max-Min Algorithm in Cloud Computing. *International Journal of Computer Applications (0975 – 8887) Volume 50 – No.12, July 2012*.

[51] Arif Mohamed. Retrieve from http://www.computerweekly.com/feature/A-history-of-cloud-computing on 22/5/2014.

[52] Asha T N, Antony P J. A Skewness Algorithm Scheduling Approach for the Energetic Distribution of Resources for Cloud Computing Environment using Virtual Machines International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064. 2012.

APPENDICES

Appendix one

```
% Matlab script for single server queuing system for cloud
computing
n = 10; % number of request in the queue
l= [15 30 45 65 85]; % average arrival rate in the queue
m= [25 50 80 95 110]; % average service rate
R = l. /m;
p0= 1.-R; % Probability of no request in the system
a=R.^n;
pn=a.*p0; % the probability that there are n request in the system
RS=R; % Mean number of request in the service facility
c=R.^2;
RQ=c./p0; % Mean number of request in the queue
TQ=RQ./l; % Time spend (delay) by a request in the queue waiting
to be served
Rs = RQ+RS; % Total request in the system
Ts=RS./l; % Time a request spent in the service facility
% The mean busy time Pb of a single server system
Tr= TQ+Ts; % Total time in the whole system (in  the queue and
receiving service)
Th= m.*R; % the Throughput of the given system
U=100.*R; % Utilization rate (%)
```

Appendix two

```
% Matlab script for Multiserver system
l=[15 30 45 65 85]; % average arrival rate in the queue
m=[25 50 80 95 110]; % average service rate n= 10;
n=10;
k=5;
c=n.*m;
R=l./c;
r=l./m;
y=factorial(k);
a=0;
for n=0:k-1
a=a+((r.^n)/(factorial(n)));
end
d = r.^k;
e= 1.-R;
h=y- e;
b=d./h;
g= a+b;
p0=1./g; % Probability of no request in the system
 %Number of request in the queue RQ and timed delay TQ
r1= p0.*d;
r2=e.^2;
r3=y.*r2;
RQ= r1./r3;

TQ=RQ./l; % The time a request spend (waiting) in the queue
% Number of request in the service facility
R1 = r;
% The Number of request in the system
R2= RQ+R1;
```

```
RT = R2./l; % The mean response time

Th=k.*l; % throughput of a complete m/m/c system

% the time in which a customer arrive a system and forces to join
a queue

k1=y.*e;

k2=d./k1;

Pn=k2.*p0;

U=100.*R; % Utilization rate (occupancy) for multiple server
```

Appendix three

```
% Matlab script for resource allocation using the proposed
algorithm

v=[512 1028 420 330 550];% Size of task (MI)

R=[128 256 284];% Processing speed (MIPS)of Resources


for i=1:5

    for j=1:3

        EET(i,j)=v(i)./R(j);

    end

end

[max1, rowIdx] = max(EET(:,1),[],1);

[rowidx,colidx]=find(EET==max1);

pivot=max1;

r(colidx)=max1;

for i=1:5

    for j=1:3

        EET(rowIdx, j)=0;

    end

end

        for i=1:5

            for j=colidx

                EET(i,j)=0;

            end

        end

 [max2,d]=max(EET(:,colidx+1),[],1);

[h,k]=find(EET==max2);

for i=1:5

    for j=3

        EET(d,j)=0;
```

```
    end
end
for i=1:5
    for j=k
        EET(i,j)=0;
    end
end
pivot2=max2;
r(k)=max2;
q=1;
max3=0;
max4=0;
max5=0;
while q==1
    for i=1:5
        for j=1:3
            if EET(i,j)>max3
                temp=max3;
                max3=EET(i,j);
            end
        end

    end
    [rowi,coli]=find(EET==max3);
        for i=1:5
            for j=1:3
                EET(rowi, j)=0;
            end
        end
                for i=1:5
                    for j=1:3
```

```
                        if EET(i,j)>max4
                            temp=max4;
                            max4=EET(i,j);
                        end
                    end
                end

            [rowi,coli]=find(EET==max4);
        for i=1:5
            for j=1:3
                EET(rowi, j)=0;
            end
        end
        for i=1:5
            for j=1:3
                if EET(i,j)>max5
                    temp=max5;
                    max5=EET(i,j);
                end
            end

        end
        [rowi,coli]=find(EET==max5);
        for i=1:5
            for j=1:3
                EET(rowi, j)=0;
            end
        end
        q=2;
    end
    if max3+pivot2<pivot
```

```
    r(k)=max3+pivot2;
    pivot2=r(k);
else
        r(k+1)=max3;
end
if max4+pivot2<pivot
    r(k)=max4+pivot2;
    pivot2=r(k);
else
    r(k+1)=max4;
end
if max5+pivot2<pivot
    r(k)=r(k)+max5;
    pivot2=r(k);
else
    r(k+1)=max5;
end
bar(r,0.6);
```