Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment

By: Alemnesh Getachew

A Thesis Submitted to the

Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfilment for the Degree of Masters of Science in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

October 2020

Adama, Ethiopia

Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment

By: Alemnesh Getachew

Advisor: Dr.-Ing Frezewd Lemma

A Thesis Submitted to the

Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfilment for the Degree of Masters of Science in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

October 2020

Adama, Ethiopia

Statement of Declaration

I hereby declare that this MSc thesis is my original work and has not been presented as a partial degree requirement for a degree in any other university and that all sources of materials used for the thesis have been fully acknowledged.

Name: Alemnesh Getachew

Signature:
_____

This thesis has been submitted for examination with my approval as a thesis advisor.

Name: Dr.-Ing Frezewd Lemma

Signature:
_____

Date of Submission: _____

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the final open defence by **Alemnesh Getachew Dubale** read and evaluated her thesis entitled **"Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment"** and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfilment of the requirement of the Degree of Masters of Science in Computer Science and Engineering.

_____	_____	_____
Advisor/Supervisor	Signature	Date
_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Acknowledgment

First of all I thank God for guiding me and taking care of me all the time.

I would like to express my profound sense of gratitude towards my advisor **Dr.-Ing Frezewd Lemma**, Department of Computer Science and Engineering, School of Electrical Engineering and Computing, Adama Science and Technology University for his continuous support, guidance, encouragement, and suggestions throughout the period this work was carried out. His readiness for consultation at all times, his educative comments have always been a constant source of motivation for me.

I would also grateful to Mr. **Endale Aregu, Mr. Anteneh Alemu,** and **Mr. Desta Zerihun** staff of Computer Science and Engineering Program in the School of Electrical Engineering and Computing, and member of cloud computing SIG for their support, helpful comments, and review contribution to the success of my thesis work.

Very special thanks to my husband Mr. **Natnael Alemayehu** who spent all his time making me happy and always the best. Thanks for your patience, for helping, guiding, and supporting me through all my life.

I want to thank my dear sisters for their encouragement and continue support. Thanks to all my postgraduate friends in Computer Science and Engineering Department.

Table of Contents

Content **pages**

List of Figures

List of Tables

List of Abbreviations and Acronyms

NIST	National Institute of Standards and Technology
RR	Round Robin
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
XaaS	Everything as a Service
SJF	Shortest Job First
VM	Virtual Machine
FCFS	First Come First Serve
QT	Quantum Time
CPU	Central Processing Unit
AWS	Amazon Web Service
QoS	Quality of Service
EDRR	Efficient Dynamic Round Robin
IDE	Integrated Development Environment
RAM	Random Access Memory
OMNT	Objective Modular Network Test-bed in c++
BT	Burst Time
DC	Data Centre
CIS	Cloud Information Service
MIPS	Million Instructions Per Second

Abstract

Recently, cloud computing has become a new global trend in computing. It's an innovative way to use the power of the Internet to provide remote resources. It is a new solution and strategy to achieve high availability, flexibility, cost reduction and expansion in demand. However, cloud computing has many challenges such as misuse of resources which has a profound effect on the performance of cloud computing. These problems are caused by huge amount of information. Thus, the need for efficient and powerful cloud computing task scheduling algorithms to improve cloud computing performance is a major issue in this field. Many researchers have proposed different algorithms for scheduling cloud computing tasks; there is still some insufficiency in system performance and low resource usage. Therefore, this study proposes a hybrid task scheduling algorithm to improve the performance and efficiency in a heterogeneous cloud computing environment. The proposed algorithm is called **Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment.** It works based on the principle of Round Robin (RR) and Shortest Job First (SJF) algorithm with dynamic quantum time. . In this work we take the advantages of both RR and SJF algorithms. We take time quantum as 80 % of the maximum burst time and a task lesser than the quantum time will be assigned in SJF manner and tasks larger than the quantum time will be assigned as their arrival time; if a new task arrives the scheduler recalculates the time quantum. The algorithm is evaluated using the CloudSim simulator and compared with RR, SJF and EDRR algorithms. As a result, we found that average turnaround time and average waiting time were minimized and performance was improved compared to other algorithms.

Keywords: Task Scheduling, Cloud Computing, Virtual Machine, Virtualization, CloudSim, Round Robin, Shortest Job First.

CHAPTER ONE

1. Introduction

1.1. Background of the Study

In recent years, cloud computing has emerged as a new computing model that has led to the rapid development of the Internet. This is leading to a new computing revolution. Cloud computing refers to a collaborative information technology environment that can be deliberately quantified with the goal of remotely distributing scalable IT resources for efficient and green utilization. It is a subsidized form of cluster computing and utility computing. It has the potential to make "computing as a utility" a reality in the future. It describes a large number of applications that can be accessed over the Internet. For this, large data servers are used to host web applications [1].

Cloud computing provides on-demand computing that highlights subscription-based services where one can obtain a huge number of computer resources and storage spaces. The clouds depict virtualized datacentres. National Institute of Standards and Technology states the cloud computing definition as, “Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [2].

According to NIST There are three service models for cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (Paas), and Software as a Service (SaaS). IaaS provides access to resources such as physical machines, virtual machines, virtual memories and virtual storage. PaaS provides a running environment for your application; it also provides the development and deployment tools needed to develop applications. The SaaS model allows software applications to be delivered to the cloud user. It refers to the software that is assigned to the hosted services and accessible via the Internet [3] [4]. Now a day, there is additional service model called everything as a Service (XaaS). This service offers everything as a service. The X in XaaS represents an unknown value, which means “everything as a service” [27].

In cloud computing there are four types of deployment models: [2]: deployment model is the way that cloud services are provided to the cloud users. These services are public cloud, private cloud, community cloud, and hybrid cloud. **Public cloud:** is a deployment model which provides systems and services to the public user. Since, this deployment model is open to everyone, it is less secure. E-mail, social media platforms are dome of the examples of public cloud. **Private cloud:** is a deployment model which provides systems and services that can be owned and managed by a company. This type of cloud is more secure because of its privacy. **Community Cloud**: is a deployment model that provides systems and services that a group of organizations can access. This type of cloud information is limited to the owner or organization. **Hybrid cloud:** is a cloud deployment model that provides a combination of two or more clouds such as public, private or community clouds. The hybrid cloud offers more flexibility than others and offers options for data delivery.

1.2. Scheduling in Cloud Computing

Scheduling in cloud computing refers to the method used for allocating tasks to virtual machines (VMs) or allocating virtual machines on available resources for satisfying user requirements [18]. The main goal of using scheduling in cloud computing is to increase the system performance and load balancing, maximizing resource usage, saving power, reducing costs, and minimizing processing time. For this reason, the scheduler must take into account the virtualized resources and the constraints required by the users to guarantee an efficient match between jobs and resources [28].

Nowadays, different types of scheduling mechanisms are available such as static scheduling, dynamic scheduling pre-emptive scheduling, non-pre-emptive scheduling, workflow scheduling, real-time scheduling, heuristic scheduling and user-level scheduling.

Static Scheduling: in this scheduling tasks are well-known. i.e. the scheduler already has a complete information of tasks, and mapping resources before execution, timely execution of task execution. **Dynamic Scheduling:** is a scheduling strategy, in which the scheduling of tasks can vary depending on the user's demand. With a dynamic schedule, cloud service providers allocate resources and remove them when needed. In dynamic scheduling cloud cannot schedule before resource usage. **Pre-emptive scheduling**: in this type of scheduling, the task can be interrupted

in the middle of its execution by removing the virtual machine from the task and reassigning it to another task. Most of the time, it is needed to execute task that has a higher priority than another task. Then, the interrupted task is pre-assigned and continued later after the priority task is completed. **Non pre-emptive scheduling**: this type of scheduling does not allow interruption while executing. A task runs till it's fully completed. **Workflow Scheduling:** a workflow is collection of tasks. The workflow structures the application as a directed acyclic graph. In this directed acyclic graph, the node represents the task and the edge represents the application dependencies. Tasks communicate with each other through dependencies. **Real-time scheduling**: In real-time scheduling, cloud end users and cloud service providers must have a promise/agreement called a service level agreement (SLA). SLA ensures that tasks or processes meet deadlines, regardless of the system load. In this scheduling priority is assigned to the scheduling of the periodic tasks with deadlines. **Heuristic scheduling:** Heuristic scheduling is a type of scheduling which tries to find optimal scheduling, but optimization is not guaranteed. The optimization problems are in the NP-hard class. Such problems can be solved using the heuristic technique. **User level scheduling**: in this type of scheduling, the cloud user buys resources on rent from the cloud service [13].

Many researchers have proposed different task scheduling algorithms whether standard scheduling algorithms as Shortest Job First (SJF), Round Robin (RR), Max-Min and Min-Min or hybrids of them as RR and FCFS (First-Come First-Serve) , RR and SJF , priority etc. Most of these existing algorithms tried to solve the task scheduling problems based on different performance metrics, but still there are some problems that the existing algorithm cannot solve. So, we need to develop an effective task scheduling algorithm to overcome the problems that occurs in scheduling.

In this research, a task scheduling algorithm called Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment has been proposed. The proposed algorithm considers an incorporation of RR and SJF algorithm with dynamic QT.

1.3. Motivation of the Study

In research world, cloud computing is becoming increasingly popular, irrespective of the fact that it is a modern concept throughout information communication technology. In cloud computing allocation of the resource extremely affects the performance of work, availability, and responsiveness. Since, the workload on the cloud are dynamic, the providers need to adopt a good scheduling algorithms to make their system fast using less resources and small response time. There are a lot of researches has been carried out in the area of task scheduling in cloud environment. Round Robin is one of the most commonly used scheduling algorithms. It is fair in that every process gets an equal share of the CPU and easy to implement. Another most common scheduling algorithm is Shortest Job First algorithm. In SJF algorithm all shorter tasks are given priority and hence scheduled first then the longer task continues. This scheduling algorithm always produces the lowest response time. These two scheduling algorithms have their limitations like: in RR the performance is depends on time quantum. Lower time quantum results in higher the context switching overhead and the higher time quantum results FIFO. In SJF algorithm the Long-burst processes are waiting until the short CPU burst tasks completed. In fact, if short-burst processes are always available to run, the long-burst ones may never get scheduled, this cause starvation for long jobs. Depending and considering these limitations of both scheduling algorithms, designing and developing hybrid scheduling algorithm by taking the advantages both RR and SJF algorithm to satisfy cloud users and to maximize resource utilizations was proposed.

1.4. Statement of the Problem

Cloud computing is a growing area that has become a business reality in information technology. However, the technology is not yet fully developed. There are still some areas that we need to focus on, resource management and task scheduling. Scheduling tasks on the server side is very difficult because the number of task requests is very large and requiring some resources to complete. Therefore, the schedule created by the server must be optimal and good enough so that each user request gets a response on time and each task gets the necessary resources to complete it [6]. Task scheduling is a major problem in the cloud environment and it leads to reduced performance of resource allocation in dynamic nature. This makes it even more difficult to send the task to resources from the queue, and another difficulty is to provide or allocating the task to

the cloud datacentre. With the advancement of technology, many Dynamic Round Robin (DRR) algorithms have been developed. In the DRR, a dynamic time quantum is chosen instead of a constant time quantum. It can be changed after a cycle or shortly after the next process arrives in the ready queue. Many DRR-based scheduling algorithms have been developed, however, the efficiency of these algorithms again depends on how a time quantum of the DRR is selected.

Round Robin: is the simplest algorithm that allocates tasks to the virtual machine for an equal amount of time. The next process take place after a fixed interval of time called quantum time (QT) [9]. In Round Robin Scheduling the performance of RR depend on the size of the QT, if the QT is too large, it cause less response time like First Come First Serve (FCFS), and if the time quantum is too small, it increases context switching which leads performance degrade for the system [7].

Shortest Job First: in this algorithm virtual machine is allocated to the task having least CPU-burst time amongst the tasks in the ready queue. User sends request to the cloud server and the will select the smallest jobs first and assign to a proper virtual machine. If there is two or more tasks having the same CPU burst time, FCFS scheduling is used i.e. one which the one arrives first, will be scheduled first by the CPU. The main problem of SJF scheduling is starvation for longer tasks, if the number of small size tasks increases the long size tasks will have to wait [11].

In this paper, we have contributed to develop an optimized hybrid scheduling algorithm with higher efficiency and lesser context switches, less average waiting and turnaround times with high throughput. We will name our proposed algorithm as “Time Optimized Hybrid Task Scheduling Algorithm”. The goal was to combine RR with SJF to minimize the response time and to select a most optimal dynamic time quantum to gain of advantage of RR algorithms.

1.5. Research Question

Based on the above problems, the study answers the following questions:

- How the Round Robin and Shortest Job First task scheduling do affects the performance of task scheduling?
- How can we improve the RR algorithm to optimize time of scheduling?
- How can the performance of the cloud scheduling with Time Optimized Hybrid Scheduling be evaluated?

To answer the above research questions and to solve the stated problems in Section 1; in the next sections, we have discussed the general and its specific objectives.

1.6. Research Objectives

1.6.1 General Objective

The main objective of this research is enhancing the cloud scheduling by using a Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment.

1.6.2 Specific Objectives

To achieve the main objective of this research, the following specific objectives are outlined:

- Review of related works on task scheduling algorithms in cloud computing.
- Study and analyze the limitation of the Round Robin and Shortest Job First scheduling algorithms.
- Design the time optimized hybrid scheduling algorithm based on the characteristics of Round robin and Shortest Job First algorithms.
- Simulate and test using performance metrics such as average turnaround time, waiting time and response time.
- Draw the conclusion and forward recommendation.

1.7. Scope and Limitations of the study

1.7.1 Scope of the study

The scope of the research includes some tasks. Time optimized hybrid scheduling algorithm crucial task that will allow us to improve the performance of the cloud system. Calculating the most optimal dynamic QT also another task. The task scheduling algorithms were developed at the datacentre level. It only manages the user request arriving at the datacentre and assigns requests based on the current performance of virtual machines. There are a lot of issues in cloud scheduling algorithms, in our work we mainly focus on optimizing the time. The performance of the proposed algorithm will be measured using simulator (CloudSim), but not real experiments.

1.7.2 Limitation of the study

This study is only concerned with optimizing the time for scheduling of user tasks to virtual machines. This does not include allocation of VM to datacentres. It is a local algorithm considers only one datacentre in one location. The proposed algorithm was implemented and tested with a cloud simulation tool, it doesn't consider real deployment environment due to the requirement of more resource.

1.8. Significance and beneficiaries of the Study

1.8.1. Significance of the study

This research study focus towards to improve the performance of RR and SJF task scheduling algorithm in cloud computing. The proposed work provide the following benefits:

- Develop an efficient hybrid task scheduling algorithm based on RR and SJF policies.
- Decide the optimum dynamic quantum time.
- Minimize the average turnaround time, average waiting time and response time.
- Provide better task scheduling algorithm.
- It will create awareness for researchers who want to extend their study on cloud task scheduling.

1.8.2. Beneficiaries of the study

The beneficiaries of the research are the following:

- ***The cloud end-users***: cloud end users can access various cloud resource through internet within a minimum time and pays as their use. A good scheduling algorithm will satisfy user requirements by improving the response time and minimizing the waiting time of tasks.
- ***The cloud providers***: providers can easily manage a cloud resource and they become more cost effective due to optimal scheduling of a task and balancing the entire load of the system.
- ***Future researchers and academicians*:** The proposed algorithm can be used by researchers and academicians for further optimization. It is also used at the university level for research and as an academic reference.

1.9. Thesis Organization

The research consists of six main chapters; the rest of the research is organized as follows:

- Chapter one describes the introduction, problem statement, objectives, scope, limitations and significances of the research.
- Chapter two discusses the detailed background of cloud computing and review of task scheduling algorithms in cloud computing environment. This section includes, the architecture of cloud computing and task scheduling algorithms, different types of task scheduling algorithms and some related and recent research works.
- Chapter Three describes the methodology of the research and strategy to identify task scheduling algorithms solutions. This section describe the research design, data collection technique, sampling techniques, data sources, issues of reliability and validity , simulation and experimental tools and types of hardware and software used to implement the proposed solution.
- Chapter Four provides the proposed solution which is called optimized hybrid of Shortest Job First and Round Robin with dynamic quantum time in cloud computing environment.
- Chapter five describes the research evaluation, results, and discussions. In this section, we analyzed the result obtained from the experiment performed and evaluating our proposed algorithm with the existing approach.
- Chapter six describe summery, conclusions and future works.

1.10. Summary

In a short in this section, we have discussed the background of the research, problem of statements, raised research questions, objectives, scope and limitations, beneficiary and significances ethical conduct. In the next chapter, we discuss technical backgrounds and related works. We also identify the gaps in different papers on the related topic to solve scheduling problems.

CHAPTER TWO

2. Technical Background and Related Works

2.1. Introduction

This chapter defines cloud computing and its characteristics, virtualization in cloud computing and task scheduling. We also discussed the challenges of task scheduling. To better understand cloud task scheduling we have reviewed a number of related works aimed at improving task scheduling in cloud computing In the next section, we look at cloud computing before we go into the details of task schedules.

2.2. Technical Background

This year's, cloud computing become very widespread as it gives greater flexibility and availability of computing resources. Cloud computing is a general term used to describe a new class of Internet based computing. It is a new model in which new technologies and existing technologies come together to serve all the capabilities of a computer system to different cloud users [8, 9].

Cloud services can be accessed from any computing devices with the help of internet connection independently of their physical location. As [15] explained about cloud computing, "it is widely spread technique which has emerged a new technology, and it provides large amounts of computing and data storage capacity to its users with a promise of increased scalability, better availability, and reduced administration and maintenance costs".

2.2.1 Cloud Computing Architecture

Cloud computing architecture consists of a front-end and a back-end that are connected through Internet. The interface includes client devices, which can be thin clients, thick clients, or mobile devices. Customers require using an interface and applications to access the cloud computing system. The backend consists of several servers and data storages. Typically, a central server manages the cloud system, monitor all traffic and efficiently meet customer demands. Cloud computing is a new form of distributed computing that provides infrastructure, platform, software and software as services. [13].

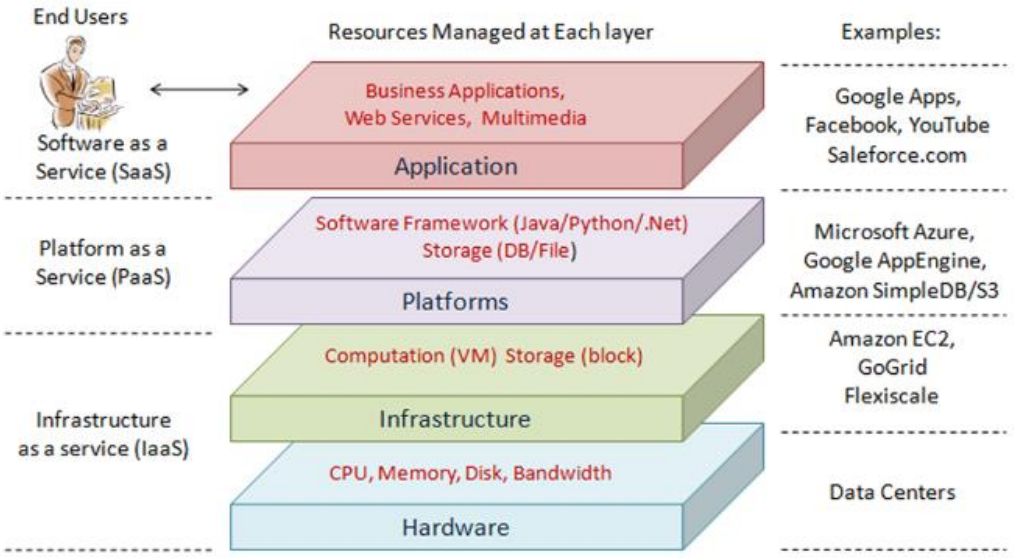

Figure 2.1 Cloud Computing Architecture [14]

2.2.2 Characteristics of Cloud Computing

The National Institute of Standards and Technology classifies five important characteristics of cloud computing. These are on demand self-service, broad network access, resource pooling, rapid elasticity and measured service [2].

- **On –demand:** Cloud service providers provide computing capabilities, like server time and network storage applications as needed without a need for human intervention with each service provider [2].
- **Broad network connection:** Cloud capabilities are accessible over the network and accessed through normal mechanisms that further use by heterogeneous thin or thick client platforms such as mobile phones, laptops and personal data assistant [2].
- **Resource Pooling:** The cloud provider pool computing resources such as storage, processing, memory, network bandwidth, and virtual machines, and these resources are dynamically assigned and reassigned according to customer's demand. Customers typically do not control or know the exact location of the resources provided [2].
- **Rapid Elasticity:** Cloud users need could service to be served rapidly. When their demand increases resources could be scaled out quickly and at once it is released it could be scaled in. The capabilities available to provide are often enormous and can be purchased in any quantity at any time [2].

- **Measured service:** Resource utilization can be tracked, monitored, and reported when consumers are charged based on their usage combination of computing power, bandwidth, and/or storage [2].

2.2.3 Cloud Computing Deployment Models

In cloud computing deployment model is a configuration of environment parameters such as accessibility of infrastructure and storage size and ownership. The type of deployment is vary depends on who control the infrastructure and its location. There are four deployment models for cloud computing. [2].

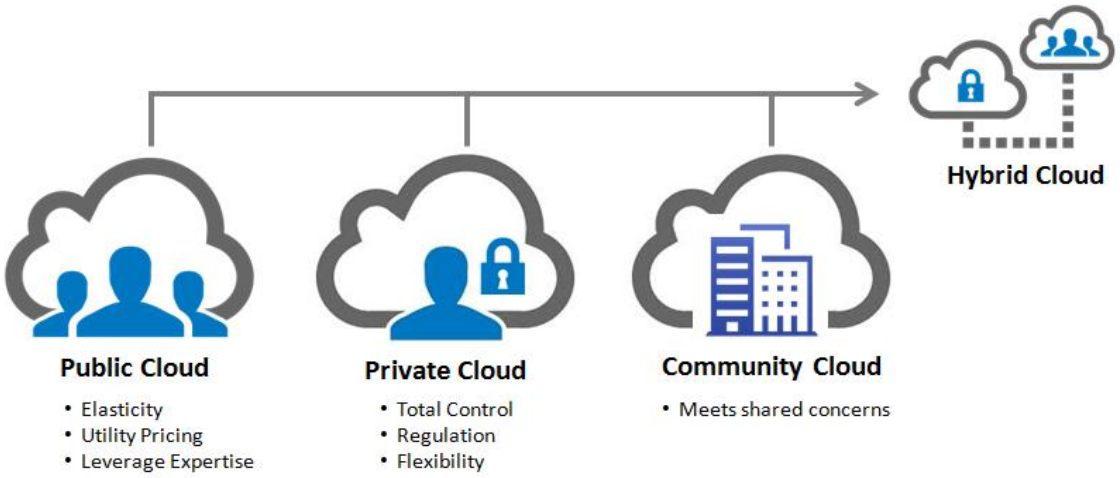

Figure 2.2 Cloud Deployment Models [44]

- **Private Cloud**

The private cloud is belonging to an organization for its various business units. It is privately managed by an organization for its own use. The organization takes advantage of the flexibility and user-friendliness of cloud infrastructure while guaranteeing the confidentiality of sensitive data. Private clouds benefit from public clouds, such as flexible and serviceable. Infrastructure may be present on or off campus and maintained in the private cloud by the organization or a third party [2].

- **Public Cloud**

Public cloud is a free cloud environment owned by a cloud provider. In the public cloud model, the cloud service provider is responsible for providing the cloud infrastructure and maintaining the user's public cloud. This means that your cloud infrastructure is accessible to the general

public or large industrial groups and is owned by the organization that markets your cloud services. Since the data is hosted in the provider's cloud, the security and privacy of user data is managed by the service provider [2].

- **Hybrid Cloud**

A hybrid cloud is a cloud environment consisting of two different cloud deployment models which are public and private. In the hybrid cloud model, cloud infrastructure provides an extension for the private cloud to bundle local resources from public cloud resources. [2]

- **Community Cloud**

The community cloud is the same as the public cloud, except that this cloud deployment model restricts cloud users' access to a specific community. This cloud deployment model is owned by either a third-party cloud provider or a community member with restricted access [2].

2.2.4 Cloud Computing Service Models

Delivering the computing resources such as operating system, hardware, software as a service rather than a product is called a cloud computing services.

Depending on the definitions of National Institute of Standards and Technology, the cloud service model is divided into three types. These are:

- Software-as-a-service (SaaS)
- Platform-as-a-service (PaaS) and
- Infrastructure-as-a-Service (IaaS).

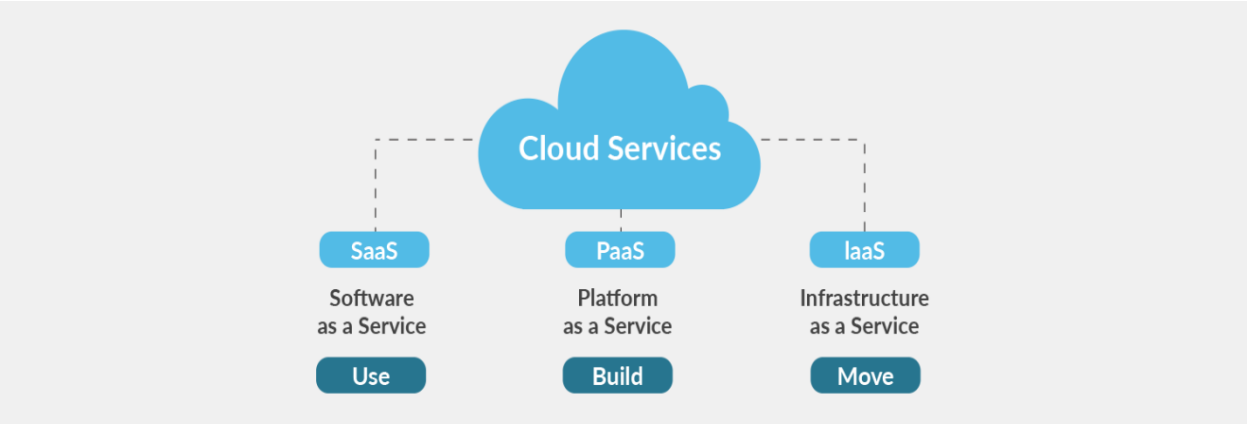

Figure 2.3 Cloud Service Models [45]

- **Software-as-a-Service (SaaS)**

Anybody can use the applications through a thin user interfaces such as web-based e-mail. Controlling the infrastructure, operating system (OS), network, storage and even the service they use and gain are not authorized for the consumers. A user will only be able to configure the application. This model eliminates most of the security threats caused by users. Software as a Service is one of the most common and popular cloud-based service models. Many companies or software providers use the SaaS model to deliver their services to the users [2].

- **Platform-as-a-Service (PaaS)**

The ability delivered to the users is to deploy on the cloud infrastructure using programming language, services, libraries, and tools encourage by the provider. Users won't have access to the underlying network, operating system, and storage. PaaS supports the cost-efficient deployment of applications without the complexity of buying and managing the underlying hardware and software layers. Even though the consumers have no control over the infrastructure, they have full control of the deployed application [2].

- **Infrastructure-as-a-service (IaaS)**

The broadly useful of an IaaS environment is to give cloud consumers with a high level of responsibility and control over its utilization and configuration. Infrastructure-as-a-Service (IaaS) specifies to offerings of necessary computer resource Infrastructure-as-a-Service (IaaS) providers that are full control of virtual resources. IaaS gives the user control over the whole infrastructure such as operating system, network, storage etc.

- **Everything -as-a-service (XaaS)**

Nowadays we can get everything as a service. Delivery of anything as a service is referred as XaaS. It recognizes the large number of products, tools and technologies to be delivered to users as a service via Internet. The X in XaaS is an unknown value, which represents "everything as a service" [27].

2.3. Virtualization in cloud computing

The cloud has an extra layer called a virtualization layer. This layer serves as a framework for building, deploying, managing and hosting application services. Virtualization means "something that is not real", but it gives all the features of a real [17]. It is a computer software implementation that will perform various programs like a real machine [20]. Through virtualization, the user can access programs or services in the cloud, so it is a core part of the cloud.

Virtualization is a combination of hardware and software engineering with which virtual machines are created and multiple operating systems can run on the same physical platform [14].

Virtualization also defined as a technique that allows logically separating of the physical resource of a server and uses them as separate machines which is called virtual machines. A single CPU becomes a multi-virtual CPU, and RAM becomes a multi-virtual RAM, and so on [23].

As [18] described, Cloud computing uses virtualization technology to map cloud resources into virtual machine layers, and to realize the task of the user so that the planning of tasks of the cloud computing environment takes place at the level of the application and the virtual resource. Virtualization enables multiple operating systems to run on the same physical platform.

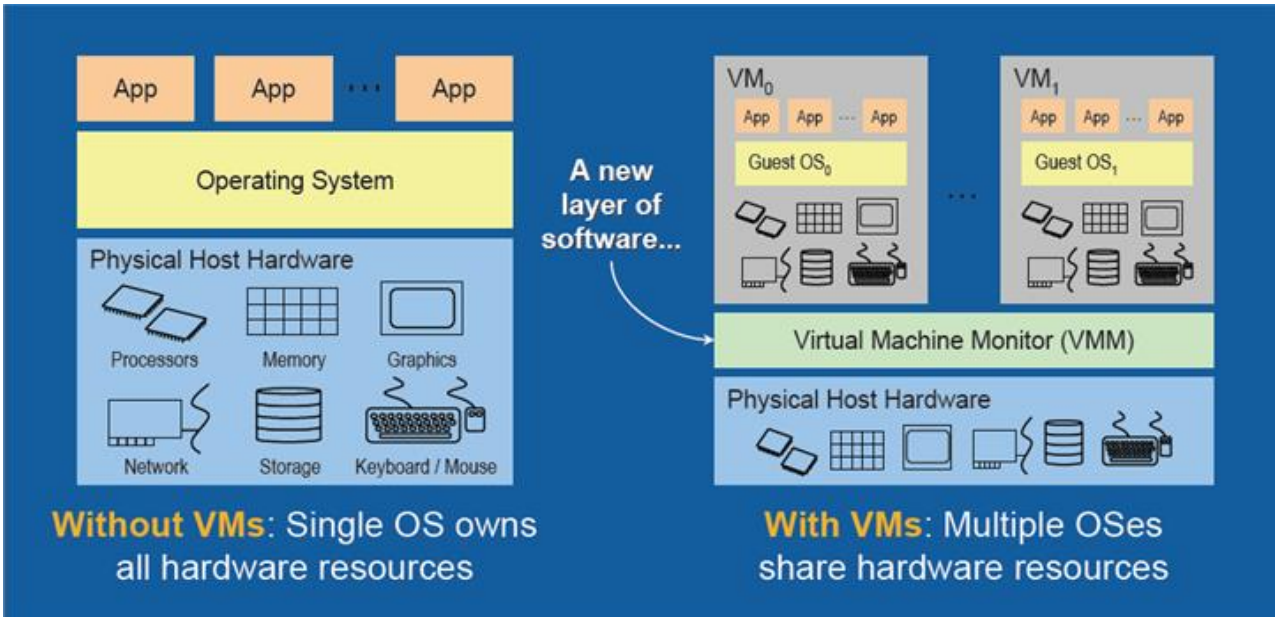

Figure 2.4 Non Virtual Machine and VM Configurations [47]

- **Type of Virtualization in cloud computing**

In cloud computing there are Different Types of Virtualization. They are:

- **Storage virtualizations**: Cloud Computing is the grouping of physical storages that are composed of various network storage devices. Storage virtualization is the pooling of physical storage devices in a way that looks like a one storage device that is managed from a central console. With the help of software applications, storage virtualization is used for backup and recovery processes [15].
- **Server virtualization:** is the partitioning of a physical server into smaller virtual servers to maximize server resources (containing the identity and number of individual physical servers, operating systems and processors). Server virtualization has a large number of benefits such as increased hardware utilization, security as well as development [15].
- **Network virtualization:** the ability to run multiple virtual networks with each having a separate control and data plan. They coexist together on top of one physical network. It can be managed by different parties that are potentially private to each other. Network virtualization create and facilitate virtual networks, logical switches, routers, firewalls, load balancers, Virtual Private Networks (VPNs), and workload securities within days or weeks [15].
- **Application virtualization:** allows the user to remotely access the application from the server. The server stores all personal information and other characteristics of the application, but can still run on a local workspace over the internet. An example of this is users who need to run two different versions of the same software [14].
- **Memory Virtualization:** with memory virtualization, physical system memory is shared and dynamically allocated to virtual machines (VMs). Virtualizing virtual machine memory is similar to the virtual memory support provided by modern operating systems [14].
- **Hardware virtualization:** is the concept of creating virtual machines (VMs) which act like a real computer with an OS. Host machine is the actual machine on that the virtualization takes place, as well as the guest machine is the virtual machines (VMs) [14].

- **Desktop virtualization**: refers to the separating the logical desktop from the physical machine. The main benefits of desktop virtualization are user mobility, portability, and easy management of software installation, updates and patches [14].

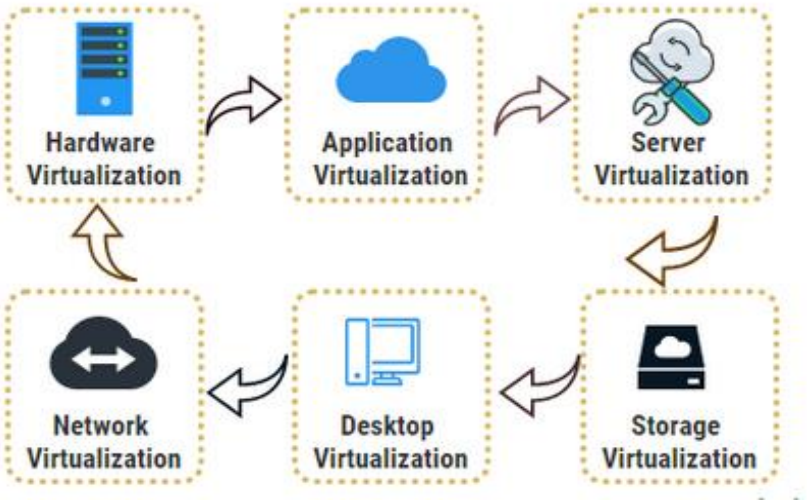

Figure 2.5 Types of virtualization [5].

2.4. Task scheduling in cloud computing.

Task scheduling is the process of arranging incoming requests (tasks) in certain manner so that the resources are used properly. Since cloud computing is a technology that provides services through the Internet, service users should submit their applications online [22].

Based on [24] research paper, the goal of cloud computing task scheduling is to provide users with an optimal task schedule and simultaneously provide system throughput and QoS. Load distribution, QoS, minimum cost, and optimal timing and system throughput are specific goals. In cloud computing systems, as the number of users increased the tasks to be scheduled in cloud increased proportionally. Therefore, we need a better algorithm to schedule tasks in these systems.

The main goal of scheduling is ensuring that the use of resources is maximized and the processing time of tasks is minimized. Schedule tasks should be ordered so that there is a balance between improving the quality of services and at the same time maintaining efficiency and legitimacy at work. An effective task scheduling strategy should aim to provide shorter response times so that submissions are executed within this time.

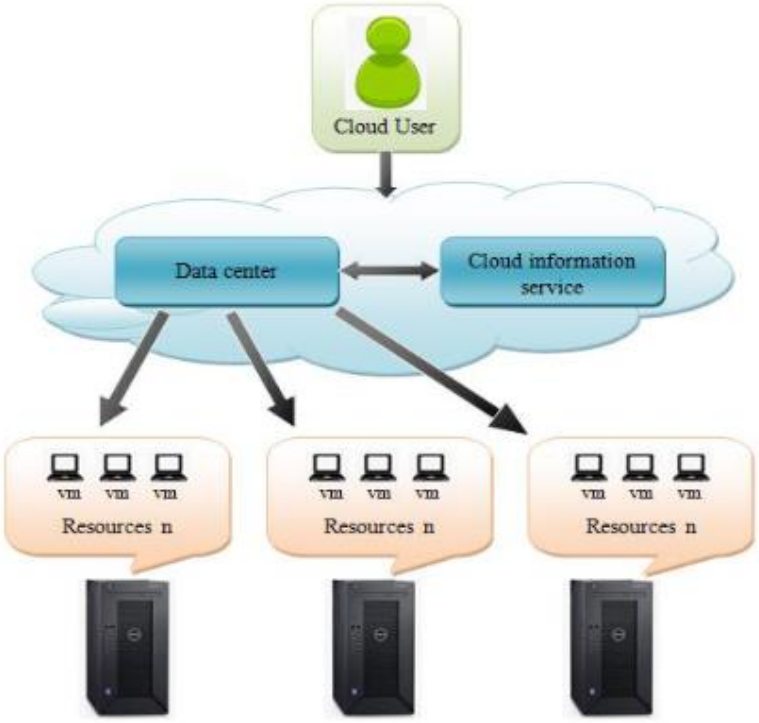

Figure 2.6 Scheduling architecture in cloud computing [29].

In the scheduling architecture, users submit their tasks to a DatacenterBroker. The DatacenterBroker acts as a dispatcher between the user and the Datacenter and helps to schedule tasks in virtual machines. The Datacenter has several hosts to which number of virtual machines is allocated, and those VM tasks are allocated according to the scheduling policies taken by the DataCenterBroker. The number of DatacenterBroker in the datacenter is similar to the number of users in the cloud [30].

Task scheduling is a major research topic designed to ensuring that every computing resources are distributed efficiently, fairly, and ultimately improves resource utilization.

2.4.1 Task scheduling algorithms

This section presents some of the most common task scheduling algorithms that are used in cloud computing environment [22]. In our work, we are going to make experiments on some of these algorithms and compare them with our work.

2.4.1.1 First Come First Served algorithm

The first-come first-served scheduling algorithm is the most basic and simple technique that uses arrival time to schedule task in a cloud environment. This work will be done exactly according to which task arrive first in the queue. It depends entirely on arrival time and does not take into account any other parameters. The task will be determined by selecting the correct task order. The task or user request that arrives at the data center first will be assigned to the VM first. This algorithm is simple and fast [31].

2.4.1.2 Round Robin algorithm

The round robin algorithm is an example of a load balancing technique. A round robin technique was developed to divide scheduling time evenly across all scheduled tasks. It adds all the tasks to the waiting list and assigns each task for equal small unit of time. RR's main concern is to distribute the load equally among all resources. The scheduler selects a task and assigned it to the controller, when the time slice for the first task has expired, proceed to the next task. The time interval between processes is called quantum time. This is the cyclical approach where all tasks are assigned to the controller at least once and the scheduler takes over the first task again. Load balancing and response time are much better compared to other algorithms. The way Round Robin works in the cloud environment corresponds to that of Round Robin in process planning. In round robin, each task has the same probability of being selected [31]. The main disadvantages of round robin scheduling are as follows [25]:

- **High Average Waiting Time**: The process is waiting in queue until his turn to own the processor. Because of the time quantum, processes are pushed to leave the processor and return to the waiting state. This procedure increase the average waiting time, which presents the main disadvantage.
- **Low Throughput:** Refers to the number of complete processes per unit of time. Because of the time slices, the round robin is characterized by a high number of context switches, which leads to overall degradation of system performance (throughput).
- **Context switch:** in RR once the quantum time expires and the process is forced to leave the CPU the scheduler stores the current process in stack and assign the CPU for next

process. This concept is known as context switching, which leads to wasted time and scheduled overhead.

- **High Response Time:** Response time is defined as the time between sending a request and the first response from the CPU. Typically, the round-robin operation increases the response time, resulting in reduced system performance. A faster response time is required to achieve higher performance.
- **High turnaround time:** it is the time between process submission and its completion. Round robin scheduling algorithm is characterized by high turnaround time. Therefore, this parameter should be reduced to improve system performance.

2.4.1.3 Shortest Job First

The principle of the Shortest Job First (SJF) scheduling algorithm is that it schedules the smallest task first. In this algorithm, the ready queue is first sorted in the ascending order of the size of the burst time of the incoming task. Then, the smallest execution time will be placed first and the longest execution time placed last and was given the lowest priority. This algorithm is optimal in most cases compared to others. But, The major challenge in SJF is large size jobs may wait longer because it has to wait not only for jobs that are in the system at the time of its arrival but also for all short jobs that are in the system at the time of its arrival. This leads starvation for larger jobs [19].

2.4.1.4 Priority scheduling algorithm

In priority scheduling, the tasks submitted to the ready queue with their assigned priorities. The operating system allocates the CPU to the task having the highest priority. The CPU is then assigned to the lowest priority task. The priority scheduling algorithm respects the priority of tasks instead of focusing on efficiency constraint of CPU. So, this may cause best and worst case depending on burst time of corresponding task [19].

Based on the advantages and limitations of these well-known cloud task scheduling algorithms, we focus on studying the round robin algorithm, which is one of the most fairly scheduling algorithm. RR is cyclic in nature, so there is no starvation for tasks. We also explore the limitations of the RR algorithm. In RR, performance is based on the quantum time, setting the

quantum too short, increases the overhead, and lowers the CPU efficiency, but setting it too long may cause a poor response to short processes, therefore setting a quantum time need to be smart. Another limitation is the average waiting time and turnaround time is often higher according to the RR. SJF algorithm has better response time and turnaround time than RR and any other algorithm, so to overcome these limitations, we came up with the idea of combining RR with SJF and determining the dynamic quantum time to optimize the scheduling algorithm. For better performance enhancement of the RR algorithm, we have improved the algorithm by minimizing the average turnaround time, waiting time, and increased response time. In the next section, we discuss the related works of various task scheduling algorithms to come up with a better solution.

2.5. Related works

Many researchers' proposed different algorithms in task scheduling in cloud computing. In this section we review a number of researches that worked on enhancement of task scheduling.

Sethi et al. [32] presented the load balancing algorithm using the fuzzy logic with round-robin (RR) algorithm. The algorithm is based on various parameters such as processor speed, virtual machine weight etc. The algorithm maintains information about each virtual machine and the number of requests currently assigned to the virtual machine. When a new request is received, the load balancer detects and assigns the task to low-loaded virtual machines, but if there are more than one virtual machine, the choice depends on the processor speed and the load on the virtual machine. The option uses fuzzy logic.

This algorithm improved the performance of the load balancer and reduced the response time. In addition, the results show that its performance is better than that of the RR algorithm. The downside of this approach is that the authors only compared their results with the RR algorithm, which has been improved and refined by many researchers in the past.

Another author of [33] proposes a new hybrid task scheduling (SRDQ) algorithm that combines the Shortest-Job-First (SJF) scheduler and the Round Robin (RR) scheduler, considering a dynamic variable task quantum. Therefore, in this article, researchers focus on calculating the optimal quantum time at each stage of the RR algorithm, they split the ready queue into two sub-queues, Q1 for short problems and Q2 for long problems using the median as the threshold of the

length of the tasks. This algorithm executes tasks interchangeably, two short tasks from Q1, and one long task from Q2 will be completed, which will reduce the wait time for long tasks without the disruption of the SJF in the selection of short tasks. The authors compared their algorithm with four other scheduling algorithms such as SJF, RR, TSPBRR, and SRSQ (time quantum= 8), the result shows that the proposed technique works better than the others. The disadvantage of this approach is it will take a long time to calculate the quantum for each round.

Article in [34] introduce an Efficient Dynamic Round Robin (EDRR) algorithm using Dynamic Time Quantum. In this work the researchers have selected the time quantum as 0.8th fraction of the maximum burst time, which is a rule of thumb. The scheduler than assigns the CPU to all processes in the ready queue, with burst time less than the quantum time while larger ones are kept a hold on. If the remaining burst time is less than or equal to 0.2 of its real time, the system allows the process to complete. As soon as all the small processes complete their execution, the time quantum is set to the maximum burst time. They compare proposed algorithm with different RR scheduling algorithm and this one perform well according to minimizing context switches ,average waiting time and turnaround time but again if the maximum burst time is more larger than all other tasks in the ready queue the scheduler will act as FIFO this will increase starvation for longest jobs. In addition this work needs to be implemented in cloud environment.

Ghanbari et al., in [35] presented a scheduling algorithm based on job priority named (PJSC) where resources are assigned to each job according to its priority. That is, the jobs with higher priority gain first access to the resources. The simulation results described by the authors indicated that the PJSC has a reasonable complexity but sufferer from increasing makespan. On top of that, PJSC may lead to job starvation, because low priority jobs can never have access to the resources they need.

The article in [36] introduces the “Efficient Optimal Algorithm of Task Scheduling in Cloud Computing environment”. The aim of this paper is to develop a generalized priority algorithm for the efficient execution of the task. The proposed algorithm is implemented and tested using the CloudSim toolkit and compared with the RR and FCFS algorithms. The result shows that the proposed algorithm is more efficient than FCFS and Round Robin algorithm. The proposed algorithm developed with limited tasks; needs improvement for a large number of tasks. The algorithm needs to be revised based on other parameters such as response time, wait time, datacenter processing time and so on.

In [37] they proposed a Modified Shortest Job First algorithm (MSJF). The aim of this work is to reduce the time it takes to complete the last task (Makespan) and minimize the average response time with maximum resource utilization. The main idea of the proposed technique is to sort the tasks in an ascending order depending on the length of the task and calculate the average of all the tasks length. Then the algorithm checks the length of each task for all tasks, if it is less than the average task length and the number of tasks in VM1 is less than the number of tasks in VM2, the task will be sent to VM1, otherwise to VM2. The authors compared their results with SJF and FCFS and the result shows that MSJF performed better than SJF and FCFS.

Ajveer Kaur, Supriya Kinger [26] study different Job Scheduling Algorithms in Cloud Computing. Shortest Job First Scheduling Algorithm is also one of them. In SJF the smallest execution time placed first and the job with the longest execution time placed last and given the lowest priority. The major challenge in Shorts Job First scheduling is long jobs may wait longer because it has to wait not only for jobs that are in the system at the time of its arrival, but also for all short jobs that are in the system at the time of its arrival. Our proposed work will take the advantage of Efficient Dynamic Round Robin (EDRR) algorithm and SJF algorithm to develop a hybrid task scheduling for cloud computing.

Table 2.1 Summary of Related Work

No	Objective	Year of Publication	Method Used and Strength	Limitation of the Study
1	Balancing load between VMs [32].	2012	Fuzzy logic: for choosing the low-loaded virtual machines and Round Robin: for scheduling tasks.	Only compared their results with the standard RR algorithm, which has been improved and refined by many researchers in the past.
2	Developing a generalized priority algorithm for the efficient execution of the task [36].	2014	Proposed algorithm perform more efficient than FCFS and Round Robin algorithm.	Tested with limited tasks; needs improvement for a large number of tasks. Need to consider parameters such as response time, wait time, processing time and so on.
3	Reducing the Makespan and minimize the average response time with maximum resource utilization [37].	2016	Checks the VM with less number of tasks and assign to them. MSJF performed better than SJF and FCFS.	The performance is improved compared to standard SJF. But, still there is starvation problem for longer tasks.
4	Overcoming the starvation problem in SJF by proposing a	2017	They used RR, SJF scheduler and consider a dynamic variable task quantum. Reducing waiting time, response time and	The quantum time calculated at each round, this will take a long time.

	new hybrid scheduling technique based on SJF and RR scheduling techniques [33].		partially the starvation of long tasks.	
5	Reduce running time of RR algorithm by using a Dynamic Time Quantum [34].	2017	RR with Dynamic quantum time. Improving performance by reducing response time and throughput increased.	if the maximum burst time is more larger than all other tasks in the ready queue the scheduler will act as FIFO this will increase starvation for longest jobs.

2.6. Summery

In summary, although many scheduling algorithms have been proposed in recent years, the above methods still suffer from certain shortcomings. As mentioned above, although round robin is the fairest scheduling algorithm which schedule tasks for equal amount of time, most of the researchers improved it by applying dynamic quantum time, but still needs improvement on minimizing the high average waiting time problem. Another method is shortest job first algorithm which schedule tasks based on the CPU burst time of the task. This method also modified by multiple researchers, they minimize the overall response time and average response time, but still there is a starvation problem for longer tasks.

Therefore, based on the above-stated gaps of the related works, we have proposed a task scheduling algorithm by using a time optimized hybrid task scheduling for cloud computing environment. In this study, we have included multiple metrics such as average turnaround time, and average waiting time. Moreover, it focuses on optimizing the algorithm for performance enhancement and overcome gaps of some researchers. Next in chapter three, we discuss methodology used for the research to solve scheduling problems for the proposed

CHAPTER THREE

3. Research Methodology

3.1. Introduction

The aim of this chapter is to outline the methodology used for achieving the objectives of this study which is introduced in the previous chapter. A research methodology is used to explain how the process of this study is carried out and systematically try to find the solution to the problem. In this section research design, data collection approaches, the proposed solution design, the proposed solution implementation and testing methods are described.

3.2. Research Design

Research is described as a way to promote knowledge enhancement. Research design is based on the concept of a designer try to find a solution related to human problem. The creation of innovative artefacts is an important part of the process, while the designed artefacts are useful and fundamental in understanding the problem.

Research design activities begin with problem formulation and continue to propose solutions. Development and Evaluation is proceeding, with the possibility of moving back to the problem formulation stage. Finally, a conclusion is drawn where the results are the final output.

To attain the objectives of the research, answer the research questions, and design our research, we have used the procedures and techniques shown as the following:

- First, we have identified the area and the problem to be solved.
- Second, we have started the research questions to be answered.
- Third, to do this, we have reviewed different articles, journals, books, and sites.
- Fourth, we have identified the required parameters to model the solution and then design the proposed solution.
- Fifth, we have prepared the experimental environment using Eclips and CloudSim tool to conduct the experiments.
- Sixth, we have documented the result that we get by using CloudSim tool.
- Finally, we have analyzed the results and evaluated the performance of the proposed method by calculating the running time of the algorithm based on our metrics in comparison with the existing methods.

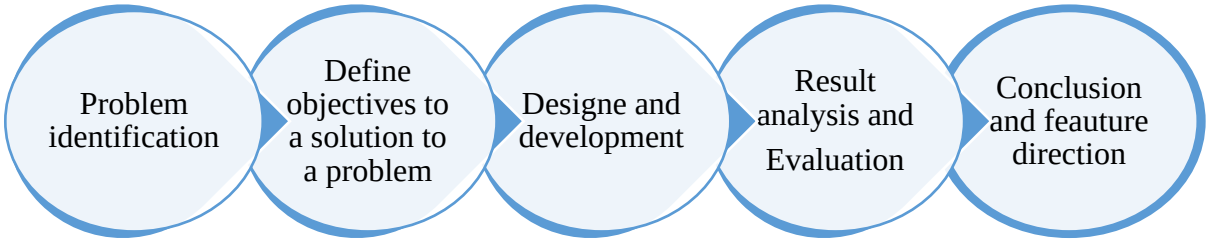

Figure 3.1 Research process

- **Problem Identification**

The problem identification is used as requirements for the development of an algorithm. Reading related and relevant research published in conference proceedings and journals are essential, which presented in the Chapter 2 (Literature Review).

Review of several algorithms which focus on task scheduling in cloud computing. These reviewed algorithms have also been used in other research areas such as load balancing, grid Computing, distributed computing etc. Clearly, identify the gap in the existing task scheduling algorithms and the parameter used for evaluation techniques.

- **Objective for solution**

The objective of our study address the tasks are needed to solve the problem of the research. Objective can be quantitative how the current one better than the existing one or qualitative how the new algorithm is expected to support solution to the problem not yet addressed.

- **Design and Development**

In this activity develop a unique architecture design, for extensibility, modularity and define functionalities of system components and interrelationships among them. After determining architecture of the new algorithm with its functionality, implement the proposed algorithm in cloudsim toolkit which runs Eclipse IDE with a different number of independent tasks and different number of VMs .

- **Result Analysis and Evaluation**

After understanding the simulation tool then implement the proposed scheduling algorithm and evaluate the performance based on different metrics such as response time, cost and processing time. Make a comparison of the proposed algorithm performance with some other task scheduling algorithm. Analyze and discuss the experimental result of the proposed task scheduling algorithm and the existing scheduling algorithm.

3.3. Data Source

For the study purpose we used both primary and secondary data sources. In this research, mostly we used secondary data source such as, articles, journals, research and other topics related to task scheduling.

- Articles\ Full text research's from early 20's to current year 2016/20 were randomly selected based on relevancy to the research
- The sources were from Google Scholar search engine and various research online databases.
    - **Primary Data**

Primary data are the data collected by a researcher specifically for a research assignment. In other word information that an organization must gather because no one has compiled and published the information in a forum accessible to the public. In this type of data gathering method, the data which the researcher collects through various methods such as interviews, surveys, questionnaires, focus groups etc. It can be also collected across the national borders through emails and posts. It can include a large population and wide geographical coverage [42].

The primary data are unique and relevant to the topic of the research study so the degree of accuracy is very high. Moreover, primary data is current and it can better give a realistic view to the researcher about the topic under consideration. Reliability of primary data is very high because these are collected by the concerned and reliable party [42].

- **Secondary Data**

Secondary data are the data collected by a party not related to the research study but collected these data for some other purpose and at different time in the past. In our research study this data become secondary data for the current users. These may be available in written, typed or in electronic forms. Secondary data is also used to gain initial perception into the research problem and classified in terms of its source either internal or external [42].

Internal data is secondary information acquired within the organization where research is being carried out. External secondary data is obtained from outside sources. Secondary data is that it is cheaper and faster to access. It provides a way to access the work of the best scholars all over the world. Secondary data save time, efforts and money and add to the value of the research study.

3.4. Data Collection Techniques

Data collection is the process of gathering and measuring the information or data from various sources for a specific purpose. And also, it enables a person or organization in order to answer relevant question and to evaluate the outcomes and make predictions about the future probabilities and trends. In the research study the data collection is defined as the systematic collection, analysis and interpretation of the necessary data to design implement and evaluate. It is also used to develop effective strategies.

In our research we used action search techniques to collect data from the internet. Websites literature reviews, previous research documents… are the main sources of data for this research in order to collect relevant user friendly design matrices. Possible information's using the recent information from the websites and CloudSim based user scalable design guidelines were selected.

3.5. Experimental Setup and Simulation Tools

In the next section, we discuss working environments that include both software requirements and hardware requirements.

3.5.1. Working Environments

Software, hardware requirements and programming language have used are discussed in the next sections.

Software Requirements

The software used to conduct this research is below.

- Operating System: - Microsoft windows 10
- IDE: - Eclipse Oxygen, NetBeans
- Programming Language: - Java
- Simulation Tool: - CloudSim

Hardware Requirements

The hardware requirement we used to conduct this research is listed below.

- Processer: Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz 2.50GHz.
- RAM: Installed memory (RAM) 4.00GB.
- System Type: x64 based processor.
- Hard Disk: 640 GB

3.5.2. Simulation tools

This section describes the simulation tools used for this research. The proposed algorithm is tested in a cloud computing environment. We have two ways to test this: the first option is to use real-time tests like Amazon EC2, and the second is to use simulation tools to simulate the cloud environment. In our work we prefer to use a simulator, as the use of real tests limits the experiments to the infrastructure level and makes the reproducibility of the results difficult [39]. Also, measuring performance in a real cloud environment is very difficult and time consuming [38]. In addition, accessing real infrastructure generates payments in real currency. In this section, we discuss some of the popular simulation tools.

- **CloudSim**

CloudSim is a development tool that provides a simple and scalable simulation framework that enables the simulation, modeling and experimentation of emerging cloud computing infrastructures and application services. The tool was developed in the Cloud Lab of the CSE Department of the University of Melbourne, Australia. It supports the modeling and simulation of various components of cloud computing, including power management, performance, data centers, computing nodes, resource provisioning, and virtual machine delivery. We need to write

a Java program to write the desired view for CloudSim and gather the results for the performance and security analysis of the cloud applications [40].

- **CloudAnalyst**

Cloud Analyst [40] is a graphical user interface simulation tool based on the CloudSim toolkit. In cloud environments, simulators can be used to test the behavior of large Internet applications during the geographic distribution of the computer server workloads and users. The main features of this simulation tool are ease of use, graphical user interface, iteration of experiments, graphical output and ease to extend.

- **Green Cloud**

GreenCloud [40] is an enhancement to CloudSim and is considered an extension of the NS2 network simulator. It is designed to validate green cloud computing. It provides detailed simulations of the power consumed by IT tools in the data center, such as network, compute servers and communication links. It is used to develop new solutions for monitoring, resource allocation, workload scheduling, and optimization of communication protocols and network infrastructure. It enables researchers to communicate, view and measure cloud performance.

- **iCanCloud**

iCanCloud [37] is a simulator tool capable of conducting large-scale experiments. This simulator is developed over SIMCAN that is used to analyze high performance I/O architectures and uses the popular OMNET++ framework for simulating computer networks. It also provides flexible, scalable, and easy to use tools. A virtual machine is the building block of the cloud computing system in this simulator. In this simulator, VM can be built by depending on its four main components and configure them. These components are CPU, memory, storage, and network. It also provides a graphical user interface for its users to configuring their simulations.

- **NetworkCloudSim**

This simulator is proposed by S.K. Garg and Rajkumari Buyya [38], an extension of CloudSim. It uses a network topology class that implements a network layer in cloudSim, reads a BRITE file, and generates a topological network. It allows for the modeling of cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations. Its structure supports designing of the real cloud data centers and mapping different strategies. It provides a

topology file which contains a number of nodes with different entities required in simulation. In NetwordCloudSim, network CloudSim work properly as every entity is mapped to a single BRITE node

- **MDCSim**

MDCSim is a discrete event simulator developed in 2009 at Pennsylvania State University, which includes hardware specifications for various servers, communications, connections, and switches that estimate their power consumption. It is a flexible and scalable modeling platform for the analysis of multi-tier data centers. In addition, it can experiment with different designs at three levels and analyze performance with a realistic workload. It accurately estimates bandwidth, response time, and power consumption [40].

Table 3.1 Cloud Simulation Tools Comparisons

Simulation Tool	**Prog Language**	**Base platform**	**GUI**	**Availability**
CloudSim	Java	SimJava/GridSim	No	Open source
CloudAnalyst	Java	CloudSim	Yes	Open source
GreenCloud	C++/OTcl	NS2	Limited	Open source
iCanCloud	C++	SIMCAN	Yes	Open source
NetworkCloudSim	Java	CloudSim	No	Open source
MDCSim	C++ and Java	CSIM	NO	Not Available

As we have discussed in above table 3.1, some of these simulation tools uses CloudSim as a base platform. CloudSim Simulation Toolkit is the most common and effective simulator for testing cloud computing-related hypothesis**.** CloudSim has some advantages: (i) Time efficiency: it requires very little effort and time to implement the testing environment and (ii) Flexibility and applicability: developers can model and test the performance of their application services with little programing and deployment effort in heterogeneous Cloud environments (Amazon EC2, Microsoft Azure. Because of these reasons we choose CloudSim to simulate the proposed solution.

3.6. Evaluation Approach

This proposed system usability evaluation is developed on Eclipse integrated with CloudSim tool. The testing can be performed during development. First, we need add some input (workload) and run the code. Finally, the system generates the result (output). Then the performance of the expected result was evaluated. There are different metrics are available for evaluating different techniques. In our work we used metrics such as Average response time, Average turnaround time, Average waiting time.

3.7. Research Methodology Development

Research methodologies can be quantitative which is based on the measurement of quantity or amount. (For example, measuring the number of times someone does something under certain conditions) or qualitative which is concerned with qualitative phenomenon involving quality (for example, asking people how they feel about a certain situation). Ideally, comprehensive research should try to incorporate both qualitative and quantitative methodologies. Research methodologies are generally used in academic research to test hypotheses or theories [48].

We have used both quantitative and qualitative research methods in which we can solve how the proposed algorithm is expected to support solutions to the problems. Quantitative research often deals with experiments, whereas qualitative research can use procedures such as field studies. We have used qualitative research to deal with the description and understanding of the situation behind the task scheduling issues and data study, as well as quantitative research for performance metrics, numerical analysis of result, and show relationships between scheduling metrics for evaluation of the study [41] [42].

3.8. Summery

In this chapter, we have discussed the research design, data sources, data collection approaches, solution design approach and tools used the proposed method implementation, evaluation methods, and the research methods. In the next chapter, we discuss the details of the time optimized hybrid task scheduling techniques.

CHAPTER FOUR

4. Time Optimized Hybrid Scheduling Algorithm

4.1. Introduction

In this chapter, we discuss the proposed algorithm for scheduling tasks across multiple computing resources which includes modelling of solution for the problem, Time Optimized Hybrid Scheduling Algorithm for Cloud Computing Environment. Furthermore, we have discussed the smarter quantum time calculation to improve average waiting time, response time and turnaround time.

4.2. Mathematical Model of Task Scheduling

In this section, we have discussed the mathematical model of task scheduling.

Let say there are **n** set of task or request need to be scheduled across the data centre given as follow:

$$T= \{T1, T2 \ldots\ldots\ldots Tn\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

And there is **m** set of virtual machine in data centre given as:

$$V= \{V1, V2 \ldots\ldots\ldots Vm\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

The current data centre load when mapping a set of user task/request **T** to the set of virtual machines **V** represents as:

$$DCL= \{VL1, VL2 \ldots. VLm\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Where

- VL1= A set of tasks mapped on V1
- VL2= A set of tasks mapped on V2
- .
- .
- VLm= A set of tasks mapped on Vm

The time needed for executing all tasks $\mathbf{t_n}$ in one virtual machine **Vi** should be calculated as below:

$$T1=\sum ti \quad i= \{1, 2\ldots. n\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

There is more than one virtual machine and a set of tasks, all tasks can be shared to multiple server nodes dealing with in parallel, the time needed **Tt** represented as below:

Tt= max {ti} i= {1, 2 …n} ……………………………….…… (5)

The objective of this research is to minimize response time, processing time and total cost including task transfer cost and virtual machine cost when assign multiple user tasks/requests to multiple cloud resources.

Response time is calculated based on the time taken DC to receive user request, the time recorded by DC and the delay time the request to reach in DC.

Rt = Ft – At + Dt …………………………………… (6)

Where
- Rt= Response time
- Ft= Finish time
- At=Arrival Time
- Dt=Delay Time

DC processing time is calculated based on user request per hour and bandwidth allocation

$$PT = \frac{R/hr}{BW} \quad \ldots\ldots (7)$$

Where
- PT = processing time,
- R/hr = request per hour and
- BW = bandwidth allocation

Task transfer cost is a cost where transfer or migrate a task from overloaded virtual machine to under loaded virtual machine.

TTC = Cost + BWcost + Tsize …………………………...... (8)

Where
- TTC = task transfer cost,
- BWcost =cost per bandwidth,
- Tsize is total task size
- Cost=0

Virtual machine cost is the cost of cloud resource based on memory cost, storage cost and processor cost.

$$VMC = Mcost + Scost + Pcost \quad \text{..... (9)}$$

Where, VMC is virtual machine cost, Mcost is memory cost, Scost is storage cost and Pcost is processor cost.

Based on the above 8&9 equation the objective function of total cost is represent as below:

$$F(TC) = \sum_{i=1}^{n} TTCi + \sum_{j=1}^{m} VMCj \quad \text{.....(10)}$$

Where
- F (TC) = Total cost
- TTCi= transfer cost of task, i= {1, 2 …. n}
- VMCj= virtual machine cost j= {1, 2 ….m}

4.3. The Proposed Algorithm

This section presents the time optimized hybrid scheduling algorithm for cloud computing environment. The simulation is implemented in the CloudSim environment which involves the host level and VM level scheduling. Proposed algorithm intends to minimize the weakness of Round Robin algorithm and maximize the strengths of both the RR and SJF by combining the two algorithms.

In this work, we try to overcome the high average waiting time of tasks problem in RR and the starvation problem of long tasks in SJF. To solve these problems, we came up with the idea of combining them together and setting a smarter dynamic quantum time. The reason for using the SJF for the hybrid is due to the fact that response time is much better than RR and using RR is for its fairness.

So in this work, we used a dynamic quantum time which is 80% of the maximum burst time, as a rule of thumb, this will help us to minimize the number of context switching in RR. The scheduler first sorts all the tasks according to their burst time in ascending order and set the quantum time as 80% of the maximum burst time, then the tasks which are less than or equal to the quantum time will be executed in SJF manner after all the tasks completed the quantum time will become the maximum burst time. Then the larger tasks will be executed in the FCFS manner. If a new task arrived the time quantum will be calculated again. The proposed time

optimized hybrid scheduling algorithm does show a better performance in this study. The process of choosing the smarter time quantum and the proposed algorithm is discussed below.

4.4. Calculating Dynamic Quantum Time

As explained above, the basis of round robin (RR) algorithm is a pre-emptive CPU scheduling that switches context after a certain time interval, regardless of the status of the process. This time interval is called the Quantum Time (QT). In such an algorithm, efficiency depends only on the amount of time set. If the time quantum is too high to burst the time of processes in the ready queue, the algorithm will only be the FCFS algorithm. Also, if the time quantum is too small, the overhead of context changes will increase, resulting in higher process latency. The efficiency of RR is somewhere in the middle. To maximize the efficiency of the RR algorithm, it is important to choose the amount of time smartly.

One of the problems with low quantum time in the Round Robin algorithm is that sometimes a process has to wait for all the other processes even if the remaining time of its implementation is very short, i.e. 1 or 2 units of time. Different researches show that, the RR algorithm produces superior performance when TQ is selected at more than 80 times of the process burst times [34].

For these reasons, we chose a time quantum as **80% of the maximum CPU burst time**. The scheduler now assigns virtual machines to all tasks in the ready queue with a burst time smaller than the time quantum in the SJF form while larger ones are kept a hold on. Also, the scheduler allows the process to run fully if the remaining run time is less than 20% of the actual time. As soon as all the smaller processes have completed their implementation, the quantum of time is set to the maximum burst time and the rest of the tasks are performed in FCFS. If a new task arrived, the scheduler recalculates the time quantum at the end of current quantum.

Our proposed approach improved the performance of the existing RR algorithm by taking the advantage of SJF scheduling algorithm.

4.5. Pseudocode for proposed algorithm

Algorithm 1: Time Optimized Hybrid Task Scheduling Algorithm

Time Optimized Hybrid Task Scheduling Algorithm

1: **Input:** User request (to be scheduled)
2: **Output:** Average Turnaround Time, Average Waiting Time, Response Time
3: initially push the processes in to the ready queue (RQ)
4: Sort all the processes present in the ready queue ascendingly.
5: **while** *(RQ != NULL)*
6: *QT=0.8* BTmax; //QT:Quantum Time, BTmax: Maximum Burst Time*
7: *If one process is there then QT is equal to BT of itself*
8: **while** *i<=N* **do** *// N: Number of processes*
9: **if** *i<N then*
10: **if** *(BT_i <= (QT)) //BT_i: Burst time of i^{th} process*
11: *Execute process for its burst time in SJF form;*
12: **else if** *(BTi > QT) then*
13: *Don't assign CPU and put the process at the end of ready queue;*
14: *Remaining Processes + +;*
15: **else if** *(i == N && Remaining Processes > 0) then*
16: *QT = BTmax;*
17: *i == 0;*
18: *i + +;*
19: If a new process is arrived update the ready queue, go to step 3.
20: **End of While**
21: Calculate Average Turnaround Time, Average Waiting Time and Response Time
22: **End**

4.6. Flow Chart for Proposed Algorithm

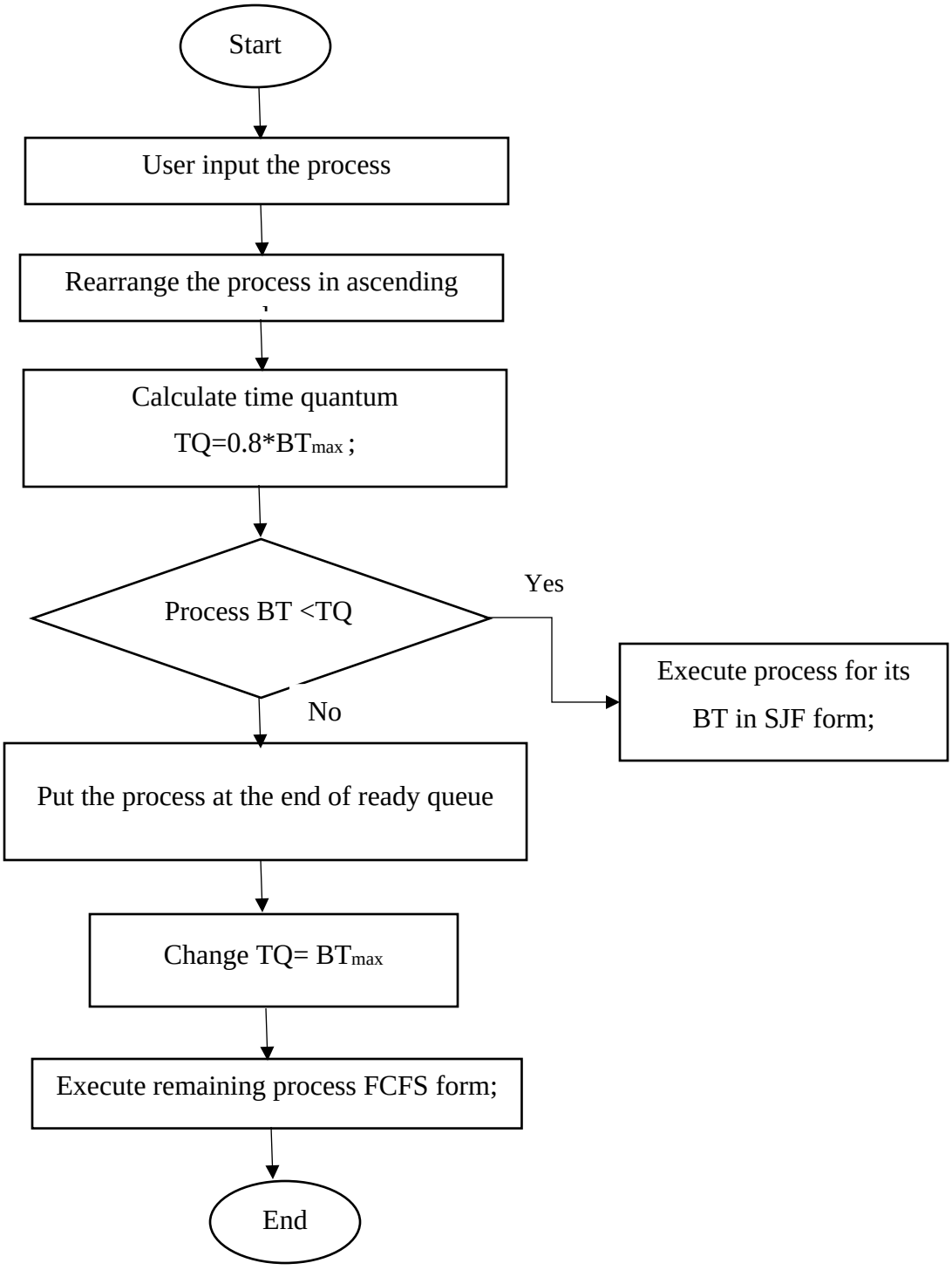

Figure 4.1 Flow chart of proposed algorithm

4.7. Summery

In this chapter, we have discussed the mathematical model for task scheduling and the proposed algorithm. We discussed in detail about how the proposed algorithm works, the concept of the dynamic quantum time and how it works also discussed. The pseudo-code and the flow chart for the proposed algorithm also discussed. In the next chapter, we will discuss the simulation of the proposed algorithm, results , comparison between the three algorithms, evaluation and discussion of the results.

CHAPTER FIVE

5. Experiment, Result, Evaluations and Discussions

5.1. Introduction

This chapter describes the environment used to perform the experiment of the proposed Time Optimized Hybrid Task Scheduling Algorithm. In addition, we will explain the simulation tool and elaborate on the results, explanations, and evaluations of the proposed method. Task schedule results are displayed using different datasets. Finally, we described the scheduling algorithm by testing our algorithm against traditional RR and EDRR. Parameters such as average processing time, waiting time, and response time are taken into account to measure the performance of the proposed algorithm.

5.2. Simulation Tools Used in the Research

5.2.1. CloudSim

It can be used to check the correctness of proposed algorithm. It is used to verify the task grouping and scheduling in simulation cloud computing environment, this simulation based on java language, supported both system and modelling of cloud computing system components (Parikh, &Sinha, 2013, and Mustafa, et al., 2014).

CloudSim [40] is a simulation tool that allows cloud developers to test the performance of their provisioning policies in a repeatable and controllable environment. This helps overcome obstacles before deploying in the real world. It is a simulator; therefore, it does not run in real environment.

CloudSim is a library for simulating cloud scenarios. It provides essential classes to describe policies for managing datacentres, computational resources, virtual machines, applications, users, and various parts of the system such as scheduling and provisioning. Using these components, it is easy to evaluate new policies controlling the use of clouds, while considering policies, scheduling algorithms, load balancing policies, and more. It can also be used to assess the competence of a strategy in terms of cost, application runtime, etc. [43].

CloudSim is a self-contained platform that can be used for scheduling and sharing policies for datacenters, service brokers, and large cloud platforms. Virtual Engine in the Datacenter

provides a virtual engine with wide capabilities for modelling structure and lifecycle management, with the flexibility of switching between core space sharing and time allocation sharing for virtual services.

The benefits of Cloudsim are (Buyya, et al., 2009):

I. Free-using (no cost) is testing service in the repeatable and controllable environment.
II. The performance is testing the bottlenecks before deploying on actual cloud.

As shown in figure (5.1), the layered is designed of the CloudSim software framework and architectural components, where is the top layer to the user code that exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirement), VMs, number of users and its application types, and broker scheduling policies (Calheiros, et al., 2011).

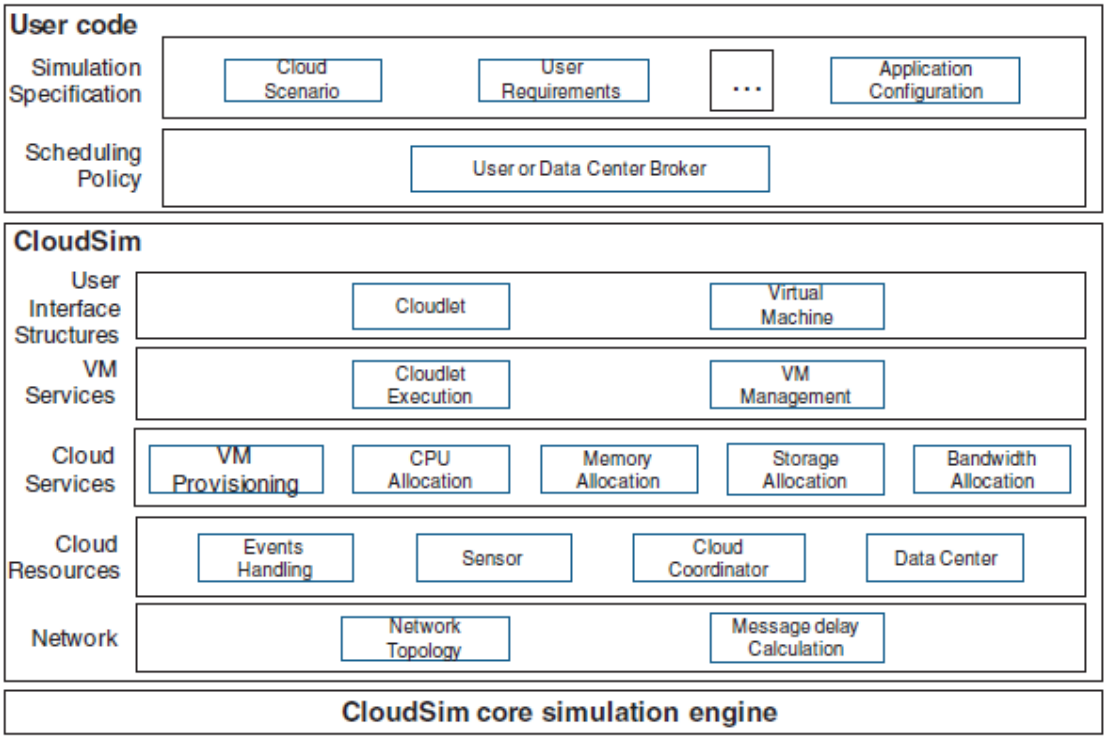

Figure 5.1 Architecture of CloudSim [44]

- The **CloudSim Core simulation engine** provides support for modeling and simulating virtualized datacentre environments, including queuing and processing of events, creating cloud objects (datacentre, host, virtual machines, brokers, services, etc.).
- The **CloudSim layer** provides virtual management interfaces for virtual machines, memory, storage and bandwidth. Additionally, it addresses other fundamental issues such as hosting hosts on virtual machines, managing application execution, and monitoring dynamic system status (network topology, sensors, storage properties, etc.).
- The **User Code layer** is a custom layer where the user writes their own code to redefine the characteristics of the stimulating environment based on their new search results.

5.2.2. Architecture of CloudSim Simulation

Below figure illustrates the main components of CloudSim, with each described below.

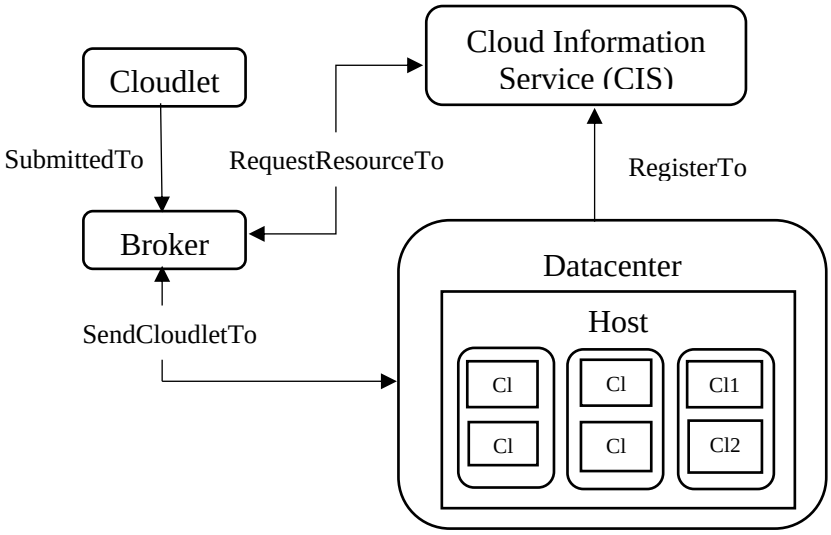

Figure 5.2 Main entities in CloudSim [43]

The above figure represents the main entities in CloudSim [43]:

1- **Datacentres (DC):** provides resource such as more than one host.
2- **Host:** physical machine which allocates more than one VMs.
3- **Virtual machine (VM)**: logical equipment on which the cloudlet will be executed.
4- **Cloud Information Service (CIS):** responsible for listing of all resources of datacentres, indexing, as well as discovering the efficiency of datacentres

5- **Broker:** responsible for an intermediate between user and service. It will send the VMs to a particular host on a particular DC and assigns cloudlets (tasks) to the particular VM. After all, the broker will destroy the free virtual machine every time the Cloudlet runs.
6- **Cloudlets:** In CloudSim, cloudlets are tasks and applications that are executed on VMs.

5.2.3. CloudSim Scheduling Policies

CloudSim has two scheduling policies: a space-shared scheduling policy and a time-shared scheduling policy. Both policies are used for VM scheduling and task scheduling. In the space-shared scheduling policy, it schedules one task to a virtual machine at a given time and another task is scheduled on a virtual machine after completion. This policy behaves like a first-come-first-serve algorithm (FCFS) [29] [30].
In the Time-Shared scheduling policy, multiple tasks are scheduled simultaneously on the virtual machine. It divided the time between all tasks and ran simultaneously in a virtual machine. The concept of a round-robin (RR) scheduling algorithm is used in this policy [29] [30].

Figure 5.3 shows the difference between these two policies and their impact on the application performance, suppose a host with two CPU cores receives request for hosting two VMs, and each one requiring two cores and running four tasks: t1,t2, t3 and t4 to be run in VM1,while t5, t6, t7, and t8 to be run in VM2.

Case 1: Space-Shared policy is used for both VM and task scheduling. Since each VM requires two cores, only one VM can be assigned to the core in a specific time. Therefore, VM2 cannot run until VM1 finishes. Also, each task hosted within the VM requires one core, so T1 and T2 will run simultaneously while T3 and T4 will wait until T1 and T2 are completed. The same happens for tasks running in VM2.

Case 2: Space-shared policy is used for VM scheduling, but time-shared policy is used for scheduling tasks. Hence, VM1 will be assigned first to the cores at a specific time and T1, T2, T3, and T4 are assigned to VM1 simultaneously. VM2 can run after VM1 finishes all the tasks. Then T5 to T8 are all assigned to it at the same time.

Case 3: Time-shared policy is used for VM scheduling, but space-shared policy is used for task scheduling. Hence, VM1 and VM2 shares a time slice of each core. Then each slice will be assigned only one task while others will wait until those tasks are completed.

Case 4: Time-shared policy is used for both VM and task scheduling. Thus, VM1 and VM2 shares a time slice of each core and all tasks share these slices simultaneously.

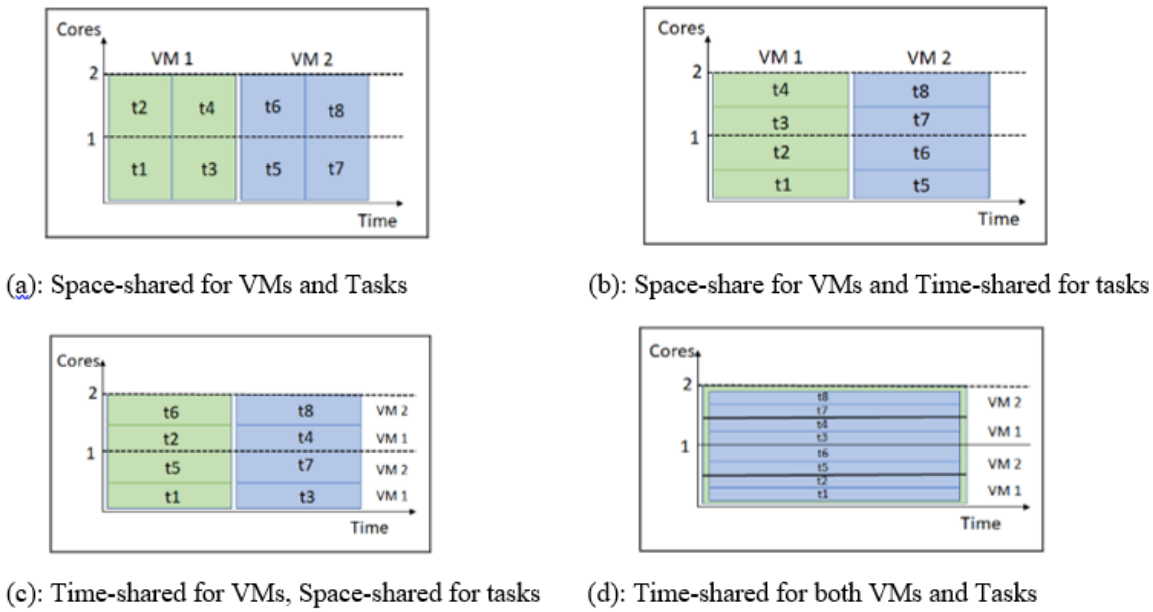

Figure 5.3 Effects of scheduling polices on task execution [29].

5.2.4. Creating and Running CloudSim

In CloudSim each simulation follows basic steps from configuration to simulation. The simulation processes are described in steps below.

1. Set the number of users for simulation.
2. Initializing the simulation by instantiating the common variables (current time, trace flag, number of users).
3. Creating Datacenters: It is resource provider and need at least one of them to run a CloudSim simulation.

   Configure of Datacenter parameters

   Datacenter (hostid,ram,storage, os, vmm, hostlist, cost …)

4. Creating Datacenter Broker
5. Creating VMs with its parameter.

   Configure of VM parameters

   VM (MIPS, id, (no. of CPU), Ram, BW…)
6. Submitting VMs to Datacenter Broker.
7. Creating cloudlets with its parameter.

   Configure of cloudlet parameters

   Cloudlet ***(***Length, id, input Size, output Size)
8. Submitting cloudlets to Datacenter Broker.
9. Sending a call to start the simulation once there is an event to be executed.
10. Sending a call to stop the simulation once there is no event to be executed.
11. Print the results of the simulation.

5.3. Experiment and result

5.3.1. Assumption

In this thesis the experiments are carried out on a single processor. All cloudlet (task) parameters, such as the number of cloudlets and burst time of all cloudlets are known before tasks are assigned to VMs. All tasks are CPU bound and none I/O bound. Context switching overhead and time taken for calculating the quantum time are not included while calculating average turnaround time and average waiting time.

5.4. Evaluation

In this thesis, we used two measurements to evaluate the performance of the task scheduling algorithm. We used Average Turnaround Time and Average Waiting Time measurement to evaluate the performance of the proposed algorithm. We performed three experiments using different datasets to compare the performance of the proposed algorithm with RR, SJF, and EDRR.

Table 5.1 Parameter used in experiment

We used the below parameters for all four algorithms RR, SJF, EDRR and proposed algorithm.

Parameter		Value
Virtual machine	Image Size	10000 mb
	memory	512mb
	Bandwidth	1000mb
	processing Speed (MIPS)	500mb
Data centre	Arch	x64
	OS	Linux
	Vmm	Xen
	DC memory	4096
	DC Storage	100000
	DC Bandwidth	100000
	DC processing Speed (MIPS)	100000
	VM policy	TIME_SHARED
	No of processor per machine	2
	QT	80% 0f the maximum burst time

- **Experiment 1 configuration**

Data Set 1: In this experiment, the application was generated in 1 data centres (DCs), 1 virtual machine (VMs), 2 dual-core host machines in each DC, and 8 constant cloudlets in the range of 1,000 to 80,000 in length, for the analysis of results.

Table 5.2 Result for Experiment 1

Represents the average waiting time and the average turnaround time for experiment 1 by using RR, SJF, EDRR and proposed algorithm.

Algorithms	Metrix	Result
RR	avgTAT	386.34
	avgWT	306.37
SJF	avgTAT	224.34
	avgWT	144.38
EDRR	avgTAT	272.77
	avgWT	192.81
Proposed Algo	avgTAT	227.89
	avgWT	147.92

Figure 5.4 Experiment one result

- **Experiment 2 configuration**

Experiment 2: In this experiment, the application was generated in 1 data centres (DCs) with the same parameter, 4 virtual machines (VMs), 2 dual-core host machines in each DC, and 15 constant cloudlets in the range of 1,000 to 85,000 in length, for the analysis of results.

Table 5.3 Result for Experiment 2

Represents the average waiting time and the average turnaround time for experiment 2 by using RR, SJF, EDRR and proposed algorithm.

Algorithms	**Metrix**	**Result**
RR	avgTAT	174.46
	avgWT	102.8
SJF	avgTAT	106.66
	avgWT	35
EDRR	avgTAT	133.87
	avgWT	62.21
Proposed Algo	avgTAT	106.66
	avgWT	35

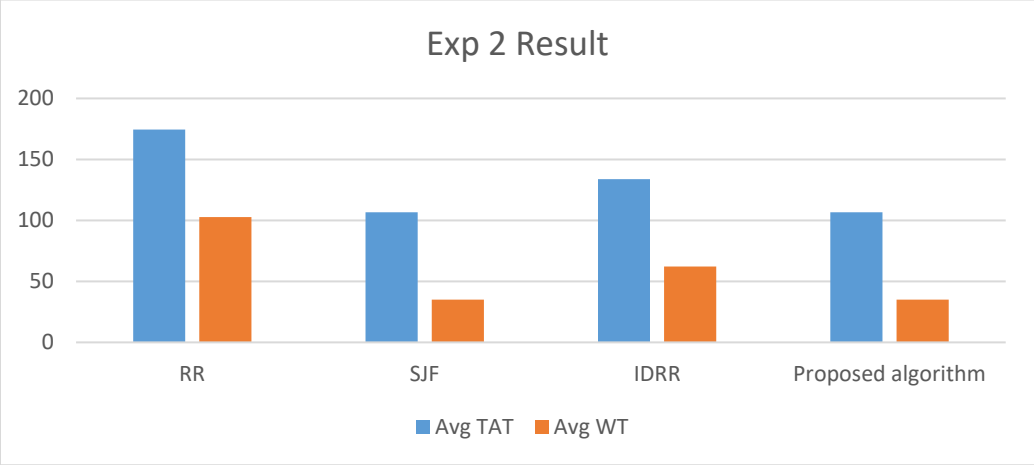

Figure 5.5 Experiment 2 result

- **Experiment 3 configuration**

Experiment 3: In this experiment, the application was generated in 4 data centres (DCs) with the same parameter, 6 virtual machines (VMs), 2 dual-core host machines in each DC, and 20 constant cloudlets in the range of 50 to 47,000 in length, for the analysis of results.

Table 5.6 Result for Experiment 3

Represents the average waiting time and the average turnaround time for experiment 3 by using RR, SJF, EDRR and proposed algorithm.

Algorithms	**Metrix**	**Result**
RR	avgTAT	96.42
	avgWT	53.87
SJF	avgTAT	62.37
	avgWT	19.82
EDRR	avgTAT	69.24
	avgWT	26.69
Proposed Algo	avgTAT	62.91
	avgWT	20.37

Figure 5.7 Experiment 3 result

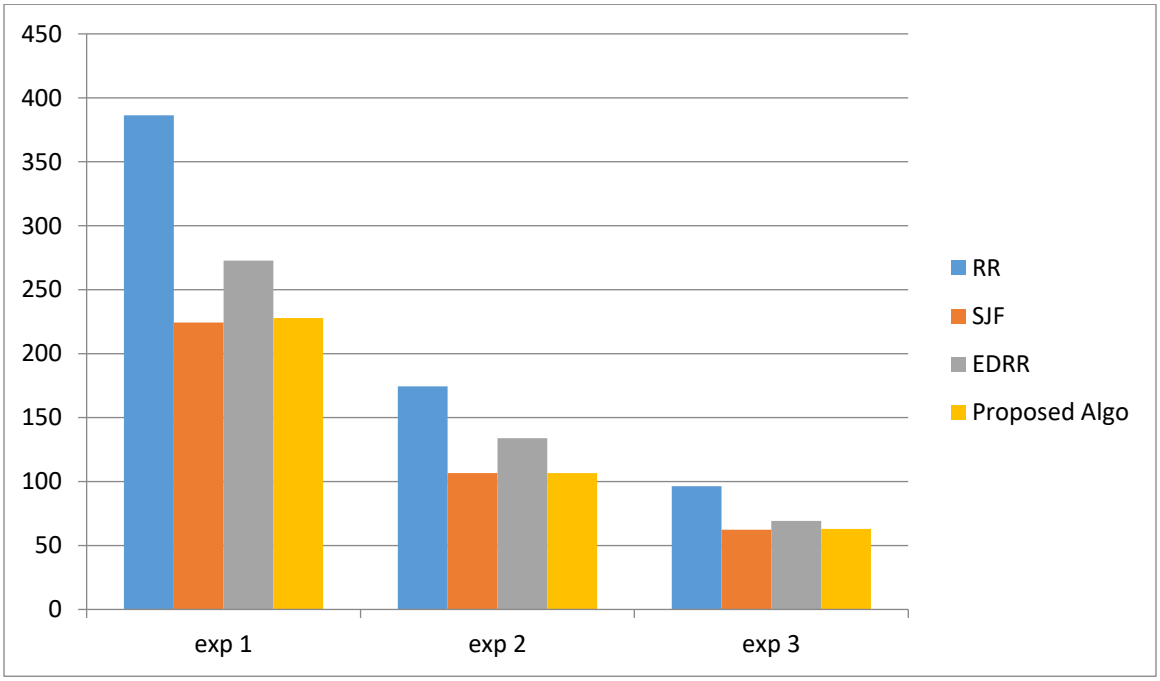

Figure 5.8 Average Turnaround time

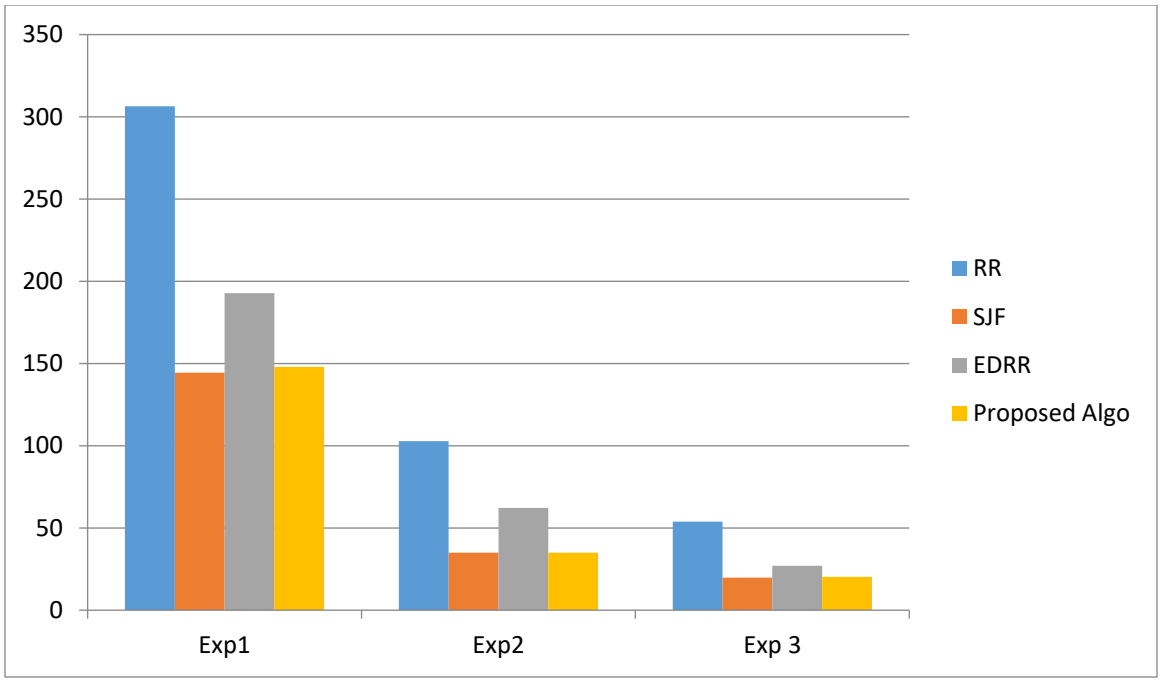

Figure 5.9 Average waiting time

5.5. Analysis and Discussion

The proposed algorithm focussed on improving the performance of the RR scheduling algorithm by combining with SJF algorithm. In this process we also come-up with the idea of dynamic time quantum which is 80 % of the maximum burst time.

We used two measurements to evaluate the qualities of service that received by the users: the average Turnaround Time and waiting time. We performed three experiments using different datasets. What we observe from the result is, when comparing the proposed algorithm with RR and EDRR algorithms, the results produced by the proposed algorithm is the lower Waiting time and turnaround time. And when comparing the proposed algorithm with SJF algorithm, the recorded result produced by the proposed algorithm is almost same to SJF without causing starvation for the longest jobs. The evaluations result of performance metrics which are shown in Table 5.2, 3 and 4 confirms that the proposed method reduced the starvation problem in SJF and reduced the waiting time problem in RR.

The three stated questions in the study are addressed based upon the result, and evaluation metrics that have been done. It is discussed as the following:

The first question raises the way in which existing RR and SJF algorithms affects the performance of the scheduling. As discussed in the related work, the existing RR algorithm affects the performance of the scheduler, if the quantum time is to short it increases context switching and if the time quantum is too long it will act as the FCFS. Due to this the average waiting time, turnaround time raise as evaluated in Table 5.2, 5.3 and 5.4. This time raise affects the performance of the scheduler. Therefore, this discussion answers the first research question.

The second raised research question is about a way of improvement of the existing RR algorithm to optimize time of scheduling. As discussed in Section 4.3, the RR algorithm is improved by combining with SJF algorithm and by calculating a dynamic quantum time. It is used to reduce the average waiting time and starvation problem for longer tasks. Here in this study, we have answered this question by proposing a hybrid algorithm. The improvement of this method is shown in evaluation section.

The third question raises the way of evaluation of the task scheduling performance. Here, we have evaluated the Time Optimized Hybrid Scheduling Algorithm performance by considering three datasets. We have used two matrixes, average waiting time and turnaround time for performance evaluation. We have compared the Time Optimized Hybrid Scheduling Algorithm with the three algorithms and the proposed algorithm performed better.

5.6. Summery

In this chapter, we discussed in detail about CloudSim simulation tool and how it works. To solve this problem successfully different simulations were done. It includes optimizing of the existing algorithm, testing of the optimized algorithm and the rest three algorithms by considering same number of tasks. Based on the result we have discussed the average waiting time and turnaround time from perspectives of the three algorithms. Lastly, we have discussed improvement of the proposed algorithm, and how researcher questions are answered. In the next chapter, we discuss conclusion and future work.

CHAPTER SIX

6. Conclusion and Future Work

6.1. Conclusion

Task scheduling is one of the important issues in cloud computing. The current task scheduling algorithms in cloud computing environment have some deficiency and this would affect the performance. Therefore we proposed a Time Optimized Hybrid Task Scheduling Algorithm to enhance the cloud computing performance. The hybrid algorithm is based on round robin and shortest job first algorithm, they take the advantages of both RR and SJF algorithms and consider the smarter dynamic quantum time to achieve the objectives. CloudSim tool used for the study, since it is the popular and comfortable tool for researchers to conduct there experiment, and freely available open source.

We set up three experimental setups and compare the results with the RR and IDRR algorithms. Analysis of the experimental results showed that the proposed algorithm gradually achieved the lowest average turnaround time, average waiting time, and higher throughput than the RR and IDRR algorithms. In the previous chapter, we used text formats, graphs, and tables to compare the results in detail. Therefore, we conclude that the proposed a time optimized hybrid task scheduling algorithm performs well in terms of average turnaround time and average wait time. Based on the simulation results, the performance of the heterogeneous cloud computing environment is improved.

The main contribution of the study is enhancing the scheduling performance by optimizing the existing RR algorithm. Studying the task scheduling in cloud environment is also another contribution. Defining the optimal dynamic quantum time is another specific contribution of the study. Moreover, improving metrics such as average turnaround time and average waiting time from perspective of the three algorithms is another known contribution of the study. So, the proposed method enhances the performance of the scheduler.

6.2. Future Work

Task scheduling is considered one of the most challenges in cloud computing, it is the major factor to improve the performance of cloud computing. The proposed algorithm is tested in a single location with the same number of cloud resources. But, still need an improvement to assign user requests to multiple location cloud resources based on other parameters and factors such as makespan time, storage cost, and memory cost, etc. The proposed solution techniques need to be implemented and applicable to real cloud service providers. In this study, we only consider scheduling issues in cloud computing, but there are other issues we are going to study in the future by using hybrid meta-heuristic algorithm.

References

[1]. hathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya, (2010).” CloudSim based Visual Modeller for Analysing Cloud Computing Environments and Applications”, 24th IEEE International Conference on Advanced Information Networking and Applications.

[2]. Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[3]. Parikh, Swapnil M., Narendra M. Patel, and Harshadkumar B. Prajapati. "Resource Management in Cloud Computing: Classification and Taxonomy." *arXiv preprint arXiv:1703.00374* (2017).

[4]. Shallal, Qahtan & Bokhari, Mohammad. (2016). CLOUD COMPUTING SERVICE MODELS: A COMPARATIVE STUDY. IEEE Network. 16-18.

[5]. https://www.educba.com/what-is-virtualization/

[6]. Varghese, Jeny, and S. Jagannatha. "Task Scheduling and VM Allocation in Cloud Computing: A Survey."

[7]. Sharma, Tejinder, Vijay Kumar Banga, (2013). “Efficient and Enhanced Algorithm in Cloud Computing.” International Journal of Soft Computing and Engineering 3(1): 385–90.

[8]. Er. Kaur, N., Er. Nagpal, P., Singh, T., (2015, Sep). “A Review on Load Balancing Techniques in Cloud Computing “International Journal for Advance Research in Engineering and Technology, Volume 3, Issue IX.

[9]. Sharma, Tejinder, and Vijay Kumar Banga. "Efficient and enhanced algorithm in cloud computing." *International Journal of Soft Computing and Engineering (IJSCE) ISSN* (2013): 2231-2307.

[10]. Farooq, Muhammad Umar, Aamna Shakoor, and Abu Bakar Siddique. "An Efficient Dynamic Round Robin algorithm for CPU scheduling." In *Communication, Computing and Digital Systems (C-CODE), International Conference on*, pp. 244-248. IEEE, 2017.

[11]. Alworafi, Mokhtar A., Atyaf Dhari, Asma A. Al-Hashmi, and A. Basit Darem. "An improved SJF scheduling algorithm in cloud computing environment." In *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pp. 208-212. IEEE, 2016.

[12]. Singh, Mohinder, and Ritu Marken. "A survey on task scheduling optimization in cloud computing." *Int. J. Adv. Res. Comput. Sci. Softw. Eng* 6, no. 5 (2016): 850-855.

[13]. Tikar, Abhijeet P., S. M. Jaybhaye, and G. R. Pathak. "A systematic review on scheduling types, methods and simulators in cloud computing system." In *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pp. 382-388. IEEE, 2015.

[14]. Kumar, Rakesh, and Shilpi Charu. "An importance of using virtualization technology in cloud computing." *Global Journal of Computers & Technology* 1, no. 2 (2015).

[15]. https://www.mygreatlearning.com/blog/virtualization-in-cloud-computing

[16]. Sajid, M. and Raza, Z. Cloud Computing: Issues & Challenges. In International Conference on Cloud. pp. 35-41, (2013)

[17]. Singh, Ashish Kumar, Sandeep Sahu, Mangal Nath Tiwari, and R. K. Katare. "Scheduling algorithm with load balancing in cloud computing." *International Journal of Scientific Engineering and Research* 2, no. 1 (2014): 38-43.

[18]. Manish Singh, R., Paul, S., Kumar, A., (2014).” Task Scheduling in Cloud Computing: Review” International Journal of Computer Science and Information Technologies (IJCSIT), Volume 5, Issue 6, pp.7940-7944

[19]. Alworafi, Mokhtar A., Atyaf Dhari, Asma A. Al-Hashmi, and A. Basit Darem. "An improved SJF scheduling algorithm in cloud computing environment." In *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pp. 208-212. IEEE, 2016.

[20]. Padhy, R.P. and Rao, G.P., Load Balancing in Cloud Computing Systems: Orissa, India, (2011)

[21]. Haque, Misbahul, Rakibul Islam, Md Rubayeth Kabir, Fernaz Narin Nur, and Nazmun Nessa Moon. "A priority-based process scheduling algorithm in cloud computing." In *Emerging Technologies in Data Mining and Information Security*, pp. 239-248. Springer, Singapore, 2019.

[22]. Shah, M. N., and Yask Patel. "A survey of task scheduling algorithm in cloud computing." *Int J Appl Innov Eng Manag* 4, no. 1 (2015): 194-196

[23]. Rashid, Aaqib, and Amit Chaturvedi. "Virtualization and its Role in Cloud Computing Environment." (2019).

[24]. Kaur, R. Kinger, S., (2014).” Enhanced Genetic Algorithm based Task Scheduling in Cloud Computing” International Journal of Computer Applications, Volume 101, Issue 14.

[25]. https://shodhganga.inflibnet.ac.in/bitstream/10603/50607/8/chap4descriptn.pdf.

[26]. Ghanbari, Shamsollah, and Mohamed Othman. "A priority based job scheduling algorithm in cloud computing." *Procedia Engineering* 50, no. 0 (2012): 778-785.

[27]. https://networklessons.com/cisco/evolving-technologies/cloud-service-models#Everything_as_a_Service_(XaaS)

[28]. Liu, Li, and Zhe Qiu. "A survey on virtual machine scheduling in cloud computing." In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2717-2721. IEEE, 2016.

[29]. Sambangi, Swathi, and Lakshmeeswari Gondi. "Task Scheduling Allocation based on Task Completion Time in Cloud Environment."

[30]. Sidhu, Harmanbir Singh. "Comparative analysis of scheduling algorithms of Cloudsim in cloud computing." *International Journal of Computer Applications* 975 (2014): 8887.

[31]. Babur Hayat Malik, Mehwashma Amir, Bilal Mazhar, Shehzad Ali, Rabiya Jalil and Javaria Khalid, “Comparison of Task Scheduling Algorithms in Cloud Environment” International Journal of Advanced Computer Science and Applications(IJACSA), 9(5), 2018.

[32]. Sethi, S., Anupama, S., and Jena, K., S, Efficient load Balancing in Cloud Computing using Fuzzy Logic. IOSR Journal of Engineering (IOSRJEN), 2(7): pp. PP 65-71,(2012).

[33]. Elmougy, Samir, Shahenda Sarhan, and Manar Joundy. "A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique." *Journal of Cloud Computing* 6, no. 1 (2017): 1-12.

[34]. Farooq, Muhammad Umar, Aamna Shakoor, and Abu Bakar Siddique. "An efficient dynamic round robin algorithm for cpu scheduling." In *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*, pp. 244-248. IEEE, 2017.

[35]. Ghanbari S, Othman M (2012) A priority based job scheduling algorithm incloud computing. Procedia Engineering 50:778–785

[36]. Agarwal, Dr, and Saloni Jain. "Efficient optimal algorithm of task scheduling in cloud computing environment." *arXiv preprint arXiv:1404.2076* (2014).

[37]. Alworafi, Mokhtar A., Atyaf Dhari, Asma A. Al-Hashmi, and A. Basit Darem. "An improved SJF scheduling algorithm in cloud computing environment." In *2016 International*

Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), pp. 208-212. IEEE, 2016.

[38]. Ray, S. and De Sarkar, A., Execution Analysis Of Load Balancing Algorithms In Cloud Computing Environment. International Journal on Cloud Computing: Services and Architecture (IJCCSA), 2(5): pp. 1-13,(2012).

[39]. Buyya, R., Ranjan, R., and Calheiros, R.N. Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. in International Conference on High Performance Computing and Simulation. Proceedings of the 2009 International Conference on High Performance Computing and Simulation, HPCS 2009, (2009).

[40]. Buyya, Rajkumar. "CloudAnalyst: A CloudSim-based tool for modelling and analysis of large scale cloud computing environments." *Distrib. Comput. Proj. Csse Dept., Univ. Melb.* (2009): 433-659.

[41]. Sileyew, Kassu Jilcha. "Research Design and Methodology." In *Text Mining-Analysis, Programming and Application*. IntechOpen, 2019.

[42]. Marczyk, Geoffrey, and David DeMatteo. *Essentials of research design and methodology*. John Wiley & Sons, 2005.

[43]. Bahwaireth, Khadijah, Elhadj Benkhelifa, Yaser Jararweh, and Mohammad A. Tawalbeh. "Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications." *EURASIP Journal on Information Security* 2016, no. 1 (2016): 15.

[44]. https://www.cloudsimtutorials.online/cloudsim-simulation-toolkit-an-introduction/

[45]. https://sciencenow001.wordpress.com/2017/03/25/cloud-deployment-models/

[46]. https://www.plesk.com/blog/various/iaas-vs-paas-vs-saas-various-cloud-service-models-compared/

[47]. https://software.intel.com/content/www/us/en/develop/articles/the-advantages-of-using-virtualization-technology-in-the-enterprise.html

[48]. Sam Goundar , Victoria University of Wellington, " Chapter 3 - Research Methodology and Research Method" March 2013

Appendixes

Appendix A: source code of optimized HRRSJF scheduling algorithm

HRRSJFDatacenterBroker.java

```
protected void submitCloudlets() {

        int vmIndex = 0;

        List <Cloudlet> sortList= new ArrayList<Cloudlet>();
        List <Cloudlet> small= new ArrayList<Cloudlet>();
        List <Cloudlet> large= new ArrayList<Cloudlet>();
        List <Cloudlet> original= new ArrayList<Cloudlet>();
        ArrayList<Cloudlet> tempList = new ArrayList<Cloudlet>();
        for(Cloudlet cloudlet: getCloudletList())
        {
                tempList.add(cloudlet);
        }
    int totalCloudlets= tempList.size();
    for(int i=0;i<totalCloudlets;i++)
    {
        original.add(tempList.get(i));
    }
                for(int i=0;i<totalCloudlets;i++)
        {
        Cloudlet smallestCloudlet= tempList.get(0);
        for(Cloudlet checkCloudlet: tempList)
```

```
                {
if(smallestCloudlet.getCloudletLength()>checkCloudlet.getCloudletLength())
                {
                                smallestCloudlet= checkCloudlet;
                                }
                        }
                        sortList.add(smallestCloudlet);
                        tempList.remove(smallestCloudlet);
        }
        long maxBurstTime = sortList.get(sortList.size()-1).getCloudletLength();
        Log.printLine("maximum= "+maxBurstTime);
        float quantumTime=0.8f * maxBurstTime;
        Log.printLine("quantum = "+quantumTime);
        for(int i=0;i<totalCloudlets;i++)
        {
                Cloudlet smallsorted=sortList.get(i);
                if (smallsorted.getCloudletLength()<=quantumTime)
                        small.add(smallsorted);
        }
        for(int i=0;i<totalCloudlets;i++)
        {
                Cloudlet largeunsorted=original.get(i);

                if (largeunsorted.getCloudletLength()>quantumTime)
                        large.add(largeunsorted);
        }
        int count=1;
        for(Cloudlet printCloudlet: small)
        {
```

```
        Log.printLine(count+".CloudlerId:"+printCloudlet.getCloudletId()+",Cloudlet
Length:"+printCloudlet.getCloudletLength());
                count++;
            }
            Log.printLine(" ");
        Log.printLine("cloudlets greater than quantum time ");
            int counttt=1;
            for(Cloudlet printCloudlet: large)
            {
        Log.printLine(counttt+".Cloudler          Id:"+printCloudlet.getCloudletId()+",Cloudlet
Length:"+printCloudlet.getCloudletLength());
                counttt++;
            }
            Log.printLine(" ");
            for (Cloudlet cloudlet : small) {
                Vm vm;
            if (cloudlet.getVmId() == -1) {
                    vm = getVmsCreatedList().get(vmIndex);
            } else { // submit to the specific vm
                    vm = VmList.getById(getVmsCreatedList(), cloudlet.getVmId());
                    if (vm == null) { // vm was not created
Log.printLine(CloudSim.clock() + ": " + getName() + ": Postponing execution of cloudlet " +
cloudlet.getCloudletId() + ": bount VM not available");
                        continue;
                    }
                }
Log.printLine(CloudSim.clock() + ": " + getName() + ": Sending cloudlet " +
cloudlet.getCloudletId() + " to VM #" + vm.getId());
                cloudlet.setVmId(vm.getId());
```

```
sendNow(getVmsToDatacentersMap().get(vm.getId()), CloudSimTags.CLOUDLET_SUBMIT, cloudlet);
                        cloudletsSubmitted++;
        vmIndex = (vmIndex + 1) % getVmsCreatedList().size();
        getCloudletSubmittedList().add(cloudlet);
                }
                for (Cloudlet cloudlet : large) {
                        Vm vm;
                        if (cloudlet.getVmId() == -1) {
                                vm = getVmsCreatedList().get(vmIndex);
                        } else { // submit to the specific vm
                                vm = VmList.getById(getVmsCreatedList(), cloudlet.getVmId());
                                if (vm == null) { // vm was not created
Log.printLine(CloudSim.clock() + ": " + getName() + ": Postponing execution of cloudlet " + cloudlet.getCloudletId() + ": bount VM not available");
                                        continue;
                                }
                        }
Log.printLine(CloudSim.clock() + ": " + getName() + ": Sending cloudlet " +cloudlet.getCloudletId() + " to VM #" + vm.getId());
cloudlet.setVmId(vm.getId());
 sendNow(getVmsToDatacentersMap().get(vm.getId()), CloudSimTags.CLOUDLET_SUBMIT, cloudlet);
                cloudletsSubmitted++;
                        vmIndex = (vmIndex + 1) % getVmsCreatedList().size();
                        getCloudletSubmittedList().add(cloudlet);

                }
                // remove submitted cloudlets from waiting list
                for (Cloudlet cloudlet : getCloudletSubmittedList()) {
                        getCloudletList().remove(cloudlet);
```

```
        }       }
```