



Department of Information Technology

**Improving Quality-of-Service in Cloud Computing Using
Hybrid Algorithms**

Prepared By

Tharaa Aljaraydah

Supervisor

Prof. Dr. Mohammad Alfayoumi

Co-supervisor

Dr. Adi Maaita

This thesis was submitted in partial fulfillment of the requirements for the
Master degree of Science in Software Engineering

Faculty of Graduate Studies

ISRA University

June 2020

AUTHORIZATION STATEMENT

Tharaa Younies Al-Jaraydah authorizes Isra University to provide hard and soft copies of this thesis to libraries for the institutions or individuals their request

Tharaa Youines Al-Jaraydah

Signature



Date:27/8/2020

اقرار تفويض

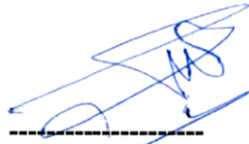
انا ثراء يونس ارشيد الجرايده افوض جامعه الاسراء للدراسات العليا بتزويد نسخ من رسالتي ورقيا و الكترونيا للمكتبات والمنظمات او الهيئات والمؤسسات المعنية بالأبحاث والدراسات العليا عند طلبها.

ثراء يونس ارشيد الجرايده

 التوقيع

التاريخ: 2020/8/27

The undersigned have examined the thesis entitled **An Improving Quality of Service in Cloud Computing Using Hybrid Algorithm**, presented by **Tharaa Youines Al-Jaraydah**, a candidate for the degree of Master in software Engineering and hereby certify acceptance.



Date

5/9/2020

Prof.Dr. Mohammad Alfayoumi



Date

5/9/2020

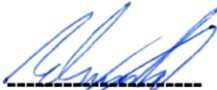
Dr. Saleh Abusouod



Date

6/9/2020

Dr. Ayad T. Al-Zobaydi



Date

6/9/2020

Dr. Adi Maaita



DEDICATION

To my dearest father Younis Al-Mashagbeh, who made me believe that a job well done is its own reward. To my precious mother Sameera Al-Omoush, who gave me the courage to be whatever I want to be. To my precious sisters and brothers, Thanaa', Roaa, Qotiaba Huthaifa, for their, support throughout my life, inspiration, and the motivation for me, who never stop giving of themselves in countless ways.

ACKNOWLEDGMENT

Firstly, I thank **Allah** for helping me complete this study. I would like to thank my instructors in the Software Engineering program at Isra University, with a special mention for my supervisor Prof. Mohammad Alfayoumi and my co-supervisor Prof. Adi Maaita for their support and guidance to improve my performance and knowledge, and for his critiques and patience which helped me continue with this study, Special thanks for DR Iyad alzobaidi for standing with me step by step to accomplish this work, and for enduring my too many questions. Honestly, I will not be able to accomplish the thesis as perfect as it is, without his support, review and corrections, and many thanks for DR Saleh abu alsoud for his instructions.

Finally, and most importantly, my family, none of this work would have been possible without them.

TABLE OF CONTENTS

Authorization Statement	Error! Bookmark not defined.
أقرار تفويض	Error! Bookmark not defined.
Dedication	V
Acknowledgment	VI
Table of Contents	VII
List of Tables	1
LIST OF FIGURES	2
LIST OF ABBREVIATIONS	3
ABSTRACT	4
CHAPTER ONE: INTRODUCTION	5
1.1. Introduction.....	5
1.2. Problem Statement.....	6
1.3. Motivation.....	7
1.4. Research Questions.....	8
1.5. Research Aims and Objectives	8
1.6. Conceptual approach.....	8
1.7. Thesis Organization	10
CHAPTER TWO: BACKGROUND AND RELATED WORK	11
2.1. Overview.....	11
2.2. Cloud Computing.....	11
2.2.1. Cloud Computing Definitions	12
2.2.2. The Architecture of Cloud Computing.....	13
2.2.3. Cloud Computing Features	13
2.2.4. Cloud Deployment Models	14
2.2.5. Service models of cloud computing	15
2.2.6. Problems in cloud computing.....	16
2.3. Scheduling in Cloud Computing.....	17

2.3.1. The Environment Features of Job Scheduling within Cloud Computing	18
2.3.2. Task Scheduling goals in Cloud Environment	19
2.3.3. Scheduling Strategies	21
2.3.4. Scheduling Process.....	22
2.3.5. Scheduling Criteria.....	22
2.4. Existing Scheduling Algorithm	23
2.5. Shortest-Job-First (SJF)	25
2.6. Related Works.....	26
2.7. Similar Approaches.....	27
2.8 Tools	28
2.9. Conclusion	29
CHAPTER THREE: HARSQ METHODOLOGY	31
3.1 Overview	31
3.2. A hybrid approach (HARSQ)	31
3.2.1. Component 1: Arrived task and arrange them	35
3.2.2. Component 2: Prepare a Queues data structure	35
3.2.3. Component 3: Modified Round Robin Algorithms	36
3.2.4. Component 4: apply SJF and check termination.....	36
3.3. Manual Case study based on HARSQ	37
3.4. Settings Simulation Environment	39
3.4.1. Visual Studio Simulation Environment	40
3.4.2. CloudSim Simulation Environment	40
3.5. Metrics Performance.....	42
CHAPTER FOUR :EXPERIMENT AND DISCUSSIONS	43
4.1. Overview	43
4.2. Prepare experimental environment	43
4.3. Comparing scheduling algorithms	44

4.4. A hybrid approach using RR and SJF by Quantum Dynamic	47
4.4.1. Component three: Modified RR	47
4.4.2. First test	47
4.4.3. Second test.....	48
4.4.4 Last test	50
4.5. A proposed Hybrid approach in C#	50
4.6. The Quality of Service Measurement	51
4.7 Discussion constructed on first dataset illustrations: -.....	53
CHAPTER FIVE: CONCLUSIONS and FUTURE WORKS	56
5.1. Conclusions.....	56
5.2. The Work of the Future	56
References.....	57

LIST OF TABLES

TABLE 2. 1 VARIOUS DEFINITIONS CLOUD COMPUTING	12
TABLE 3. 1: ARRANGE TASK BASED ON BURST TIME AND ARRIVE TIME (SJF).....	35
TABLE 3. 2: A RESULT CASE STUDY.....	36
TABLE 3. 3: COMPONENT 1 ARRIVED TASKS.....	37
TABLE 3. 4: COMPONENT 1 ARRANGE TASKS ACCORDING TO SJF	37
TABLE 3. 5: COMPONENT 2 DIVIDE QUEUE FOR TWO QUEUES	37
TABLE 3. 6: COMPONENT3 APPLY RR (QUANTUM CALCULATIONS ROUND1)	38
TABLE 3. 7: COMPONENT3 APPLY RR (QUANTUM CALCULATIONS ROUND2)	39
TABLE 3. 8: COMPONENT 4 THE RESPONDING, WAITING AND TURNAROUND TIMES BY HARSQ COMPARED TO TRADITIONAL SJF AND RR.....	39
TABLE 4. 1: DATASET USED ON EXPERIMENT (COMPONENT 1)	44
TABLE 4. 2 THE EVALUATION RESULTS OF THE SIX IMPLEMENTED ALGORITHMS	45
TABLE 4. 3 FIRST TEST DATASET	48
TABLE 4. 4 SECONED TEST DATASET	49
TABLE 4. 5 TEN CLOUDLETS BY RANDOM LONG BURST AND ARRIVAL TIME PRODUCED BY THE ENVIRONMENT.....	50
TABLE 4. 6 FIRST EXPERIMENTATION RESULTS USING 3 DIFFERENT VM.....	52
TABLE 4. 7 SECOND EXPERIMENTATION RESULTS USING 3 DIFFERENT VM.....	53

LIST OF FIGURES

FIGURE 1.1 CONCEPTUAL APPROACH	<u>9</u>
FIGURE 2. 1 CLOUD ARCHITECTURE REGARDING CLOUD SERVICES	<u>14</u>
FIGURE 2. 2 CLOUD ARCHITECTURE REGARDING (PATEL ET AL., 2012)	<u>15</u>
FIGURE 2. 3 ORIGINAL ROUND ROBIN ALGORITHM (YASSEIN ET AL., 2013)	<u>24</u>
FIGURE 2. 4 SJF ALGORITHM (ALWORAFI ET AL., 2017)	<u>25</u>
FIGURE 3. 1 HARSQ ALGORITHM FLOWCHART	<u>32</u>
FIGURE 3. 2 CLOUDSIM SIMULATION PARAMETERS	<u>41</u>
FIGURE 4. 1 PREPARE EXPERIMENTAL ENVIRONMENT WITH C#	<u>43</u>
FIGURE 4. 2 RESULT OF TIME USING SIX SCHEDULING ALGORITHMS.	<u>45</u>
FIGURE 4. 3 REPRESENTS THE EVALUATION RESULTS OF THE SIX IMPLEMENTED ALGORITHMS	<u>46</u>
FIGURE 4. 4 COMPARISON RESULTS FOR FIRST TEST	<u>48</u>
FIGURE 4. 5 SECOND COMPARISON RESULT	<u>49</u>
FIGURE 4. 6 DIFFERENT THROUGHPUTS ACHIEVED	<u>51</u>
Figure 4. 7 Result of testing HARSQ on 3 different VM.....	<u>52</u>
Figure 4. 8 Second Experimentation Results using 3 different VM.....	<u>54</u>

LIST OF ABBREVIATIONS

#	Abbreviation	Full Expression
1	SaaS	Software-as-a-Service
2	QoS	Quality of Service
3	HARSQ	Hybrid Approach Round Robin Shortest Job First with Dynamic Quantum
4	IOT	Internet Of Things
5	FCFS	First Come First Serve
6	SOS	System Of System
7	R.R	Round Robin
8	SJF	Shortest Job First
9	NIST	National Institute Of Standards and Technology
10	GUI	Graphical User Interface
11	PAAS	Plat Form As A Service
12	IAAS	Infrastructure As A Service
13	VPC	Virtual Private Cloud
14	IBM	International Business Machine
15	CPU	Central Processing Unit
16	API	Application Programming Interface
17	FIFO	First In First Out
18	DRR	Dynamic Round Robin
19	RRR	Randomized Round Robin
20	SRTF	Shortest Remaining Time First

ABSTRACT

Cloud computing is virtual environment that deliver many services and applications via internet. Quality of service (QoS) is one of the main factors that improves the performance of cloud computing. There are various techniques that address improving quality of service, yet there is no clear option that would work more efficient than the other techniques.

To ensure quality of service in cloud computing, this research we propose a novel hybrid task scheduling algorithm named (RSDQ) based on tow of shortest-job-first and round robin schedulers using a dynamic variable task quantum considering splitting the ready queue into two sub-queues, Q1, and Q2. Allocating tasks to resources from Q1 or Q2 are done mutually two tasks from Q1 and one task from Q1. The proposed algorithm was implemented in two different environments C# and CloudSim where the experimentations results and tests proved that the proposed algorithm had improved the average waiting and response times and also partially reduced the starvation over the state of art algorithms.

Keywords: Cloud Environment, Cloud Computing, Quality-as-a-Service. Load Balancer, Software-as-a-service (SaaS), round robin, shortest job first.

CHAPTER ONE: INTRODUCTION

1.1. Introduction

Currently, Task Scheduling is considered to be a prominent problem in cloud computing. It allows interdependent tasks to be mapped to virtual machines (VMs), completing the task execution within a specified Quality of Service (QoS) requirements (S. Kaur et al. 2019). Cloud computing is basically Infrastructure as a Service (IAAS). The resources in cloud computing are delivered over the network as a service (Manvi and Shyam, 2014)

The growing usage of the Internet has facilitated by the use of cloud computing in all fields of academic, industrial and social life, enabling service providers to provide services which have functional and non-functional features which available to users upon request and as needed (Jula et al., 2014).

Cloud computing has many advantages, including speed, flexibility, and opportunities to save costs since it allowed the user to use resources, internet, storage, platform, infrastructure. This thing encouraged many companies to provide cloud services through them, such as Amazon, Google, and Salesforce (Zhang *et al.*, 2010) provide many advantages for cloud computing. Cloud computing is a simulated environment that can provide many service applications through the Internet (Wickremasinghe *et al.*, 2010).

Quality of services (QoS) is one of the main factors that improve the quality of cloud computing, Cloud computing can guarantee service quality to users in terms of hardware CPU performance, bandwidth display, and memory capacity, and how to progress the performance of QoS that improve the quality time of the implementation of services. (Azeez *et al.*, 2010).

Technology-controlled application management manages scheduled applications and services to manage service time to provide service quality. This is based on the reason that application management allocates resources to ensure that services depend on execution time (Demchenko *et al.*, 2011). Various technologies are committed to improving the service quality. However, there is no clear choice that is more effective than other methods. A load scheduler determines which task will be executed before completing another task through a series of algorithms. There are two scheduling used in cloud environments, which can

independently perform tasks and services in cloud computing (Buyya *et al.*, 2009). The objectives of this thesis are to examine various technologies or algorithms to build a new algorithm and make a comparison between them to choose a suitable algorithm, we will use the modified Round Robin (RR) and Shortest Job First (SJF) to get a balance between the waiting time and response time.

1.2. Problem Statement

Cloud is a platform that allows users to share resources, suitable for multitenant computing. Scheduling uses an effective task organizer in cloud computing to search, schedule, and then assign resources for users based on user requirements (Dave *et al.* 2015).

The main problem in cloud computing is scheduling because cloud providers must provide services to many users in cloud computing systems. Therefore, the schedule is the main problem in setting up a cloud computing system (Lakhani *et al.*, 2013). Cloud computing's efficient job scheduling is affected by enhanced operations and applications of resources, which leads to high performance. The problem of task scheduling can be solved by many strategies, such as prioritization strategy, which can use some dynamic or static methods established by static or dynamic scheduling methods (Lakhani *et al.*, 2013)

The problem under service quality is how to have an application for managing allocated resources in order to protect services based on performance, accessibility, and reliability (Zhou *et al.*, 2016). Jelassi *et al* (2017) analyzed them, focusing on the investigation and analysis of technologies that affect service quality, the researchers proposed a performance monitoring application that uses System of Systems (SoS) to monitor and enhance service quality in the cloud environment (Jelassi *et al.*, 2017).

Cloud computing has a significant part within our lives and has positive effects in terms of sharing resources and accessing it quickly and updating data and is a focus of attention for designers, developers, and consumers. Designers and developers are free to experiment and develop. For the consumer all the time, the service is available to them (Attaran *et al.*, 2019). Simultaneous is limitless and allows users to share and provide Knowledge. Cloud computing performance depends on job scheduling algorithms (Jelassi *et al.*, 2017); task scheduling depends on the queue concept, in terms of waiting time. This thesis provides a new face of hybrid algorithms that had been applied to enhance the performance of cloud

computing by achieving the balance between waiting time and response time using HARSQ approach. The proposed approach is trying to avoid some problems such as long waiting time and long response time that caused from the starvation problem seriously affect the development of cloud computing service quality.

1.3. Motivation

In order to find the best kind of task scheduling is very important in cloud computing. It is important to determine the beginning and end time of the different tasks that have some limitations such resource limits or time restrictions may be imposed. The task planning aspect of cloud computing is very critical. Task scheduling helps optimize resource usage by distributing load over multiple processors and reducing the running time. There are two types: static planning and dynamic planning. The starvation issue has been one of the biggest problems facing cloud task scheduling, with the task waiting a long time for one or more of its resources. Starvation is often caused by errors in the measurement of schedules and by misuse of resources and could be intentionally generated by denial of operation (Kurdi *et al.* 2014).

The motive behind this thesis is to try to find out a novel approach by which the starvation problem can be solved. It is noted that though Round Robin (RR) was widely applied in the hybrid techniques to resolve the starvation problem such as First Come First Serve (FCFS)&RR, and Shortest Remaining Time First (SRTF)&RR (Santra *et al.* 2014). but none of them could solve the problem. It was found that the hybrid of SJF and RR ,were the furthestmost influential hybrid to solve starvation (Alworafi *et al.*, 2019). The SJF performance help reducing turnaround time and RR help in dropping task waiting time. It was also found that there was not a suitable methodology to be used, the drawbacks of the RR time quantum is static so as to estimate task quantum value because including a small quantum result in reduction throughput. In contrast, an increase in time while including long quantum leads to a rise in turnaround time.

In this thesis we present a new hybrid scheduler approach including SJF and RR with dynamic quantum, implemented by two queues that schedule processes for implementation. Also, we admit the importance of the starvation problem in task scheduling. The proposed approach is designed to be a component-based algorithm that effectively queues and optimizes the execution time. The proposed technology determines the static and dynamic

value of time to detect the effect of quantum dynamics on starvation problem in cloud task scheduling and the decrease of response time.

1.4. Research Questions

- 1) Do some of hybrid algorithms achieve an efficient quality cloud computing service process?
- 2) How to address and control starvation problem over task scheduling?

1.5. Research Aims and Objectives

Different cloud computing providers use different load balance. The objective of this thesis is to initiate a study that will provide new method or technique for improving quality of service, using a new approach of hybrid algorithm called “Hybrid Approach Round Robin Shortest Job First with Dynamic Quantum (HARSQ)” to enhance the quality of service by achieving balance among arrived time and turnaround time (Lakhani *et al.* 2020).

1.6. Conceptual approach

Proposed scheduling approach is essential that can reduce the waiting time, consumes less starvation and improves the resource utilization of the resources so speed up the response time. The approach proposed in this thesis will focus on the optimization based scheduling. By utilization RR and SJF. The approach is illustrated in Figure 1.1.

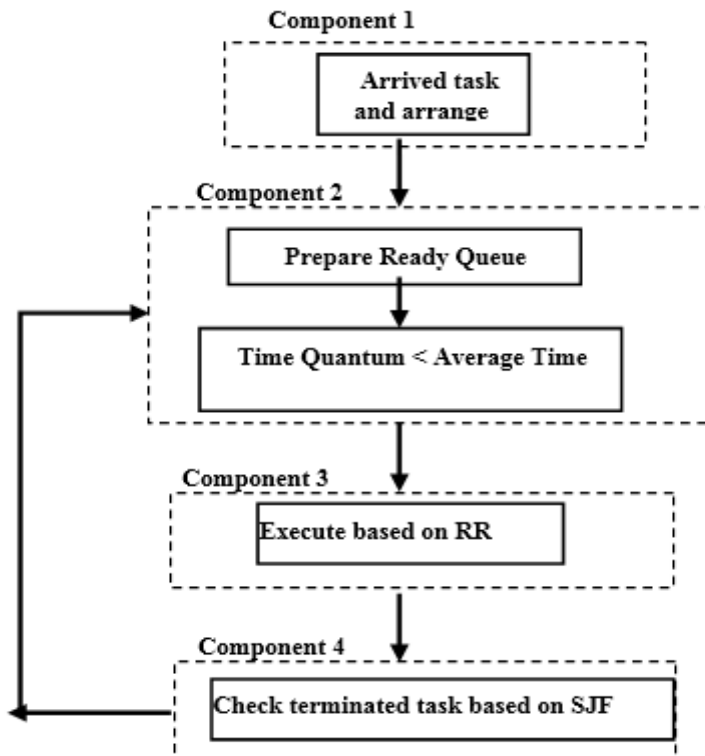


Figure 1. 1 Conceptual approach

1.6.1. Component 1

In this component the random task arrived and arrange them based on the burst time if the burst time equal the arrangement process become based on arrival time.

1.6.2. Component 2

In this component calculate the average of the burst time to prepare a ready queue if the burst time of the task is less than the average add it into Q1 otherwise add it into Q2.

1.6.3. Component 3

Applying modified Round Robin Algorithm that have a dynamic quantum time depend on task source queue.

1.6.4. Component 4

Last component applying the SJF and check the termination of task.

1.7. Thesis Organization

Chapter two is a background of cloud computing; it tackles the following issues: definition of the term, architecture, characteristic and deployment of cloud computing model, and finally a discussion of the challenges encountering cloud computing. It afterwards provide an overview of task modeling presenting its characteristics. It then switches to target of scheduling in a heterogeneous environment providing guidelines. All such things will enlighten users concerned with cloud scheduling.

Chapter three will shed light on the method that the researcher proposes through example, results are verified via the use of CloudSim simulator and C#.

Chapter four this chapter outlines of such experiment in the form of tables and figures.

Chapter five it summarizes weakness and strengths of a whole as manifested in the conclusion and the recommendations the research presents for future studies.

CHAPTER TWO: BACKGROUND AND RELATED WORK

2.1. Overview

This chapter will focus on explanations of the terms required to understand the challenge, issues relevant to cloud computing; its characteristics, types, and computing service models of cloud computing. Furthermore, it contains 2 main sections. The first section, introduces more formally and precisely the notions of keywords, the second section discusses the previous work related to the field of cloud computing quality of service and finds a suitable component to achieve a hybrid approach using to make a balance in task scheduling for improving quality of service.

2.2. Cloud Computing

Cloud Computing can be defined as a computing service built on the Internet, through which hardware, software, and data sharing to meet the user requirements. This is mainly manifested in the growth of services generated by the Internet, the use of the Internet for practice and the delivery of information to dynamically provide accessible virtualized resources (Kaur and King 2014).

Cloud computing, which is the technology of applying virtualization, can benefit from this technical support virtualization. During the last few years, many types of research were done by international enlargement companies, Like Google group, Microsoft, Alisoft, and Amazon. etc., approximately all world around companies In addition to academicians initiated a search using cloud computing technologies with related theory in using the cloud (Xu, 2012)

Cloud computing companies IT infrastructure delivery as well patterns use regarding Large Internet companies like Amazon, Netflix, and LinkedIn are using the service architecture pattern to deploy large applications in the cloud as a set of services that can be developed, tested, implemented, scaled, operated and upgraded independently. However, independent development, and scalability, infrastructure costs are a significant concern for companies adopting cloud (Villamizar *et al.*, 2016).

2.2.1. Cloud Computing Definitions

Many descriptions intended aimed at cloud computing investigated by previous works; some definition doesn't explain all cloud features. Vaquero *et al* (2008) tried to reach a universal meaning by various respects (Jula *et al.*, 2014). Researchers have attempted for description cloud technique. National Institute of Standards and Technology (NIST) characterized the cloud as shown through Table 2.1 that provides standards definition of cloud computing (Mell and Grance, 2011).

Table 2. 1 various definitions cloud computing

Source	Definition
NIST	“Cloud computing is a model for enabling ubiquitous, 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. This cloud model is composed of five essential characteristics, three service models, and four deployment models” (Mell and Grance, 2011).
Gartner	“a style of computing in which massively scalable IT-related capabilities are provided “as a service” using Internet technologies to multiple external customers”(Brodkin 2008)
IDC	“an emerging IT development, deployment, and delivery model, enabling real-time delivery of products, services and solutions over the Internet (i.e., enabling cloud services)” (Böhm et al., 2011)
Merrill Lynch	“The idea of delivering personal (e.g., email, word processing, presentations.) and business productivity applications (e.g., sales force automation, customer service, accounting) from centralized servers” (Jelassi <i>et al.</i> , 2017)

The cloud has been a systematic notion of technology stand up in few years of reality; as, in contrast, John McCarthy suspects that its infrastructure is very groundbreaking because it makes consistent computing confidence a "utility tool." Based on NIST (Demchenko *et al.*, 2011), Cloud computing consists of 5 chief features, and two other features have been added

to the previous works, including three cross- models of service then four models of deployment, that would be explicitly labelled in the next sectors (Jula *et al.*, 2014).

2.2.2. The Architecture of Cloud Computing

The architecture of cloud computing was separated into three independent strata which are; operator scale, middleware, and system layer (Patel *et al.*, 2012). These layers depend only on outputs in the lower layer. Cloud user-level includes GUI which will be in favor of the user who can forward it into upper-scale; the central middleware of specific software systems such as NetBeans Eclipse or any kind of the compilers (Patel *et al.*., 2012). This assists the user by offering them a chance to write down their software and used that on the cloud provider. The system-level control needs to be controlled as that allows the user to have control over all providers' resources. Due to that, users can make different types of changes the way they like as they have control over the lower level and hence the development is easily done.

2.2.3. Cloud Computing Features

The main Features for Cloud Computing contains (Xu, 2012):-

- 1- *On-demand self-service* registration probably a resource could computing be obtained and utilized without any time to establish interpersonal relationships through cloud management suppliers. Resources of computing consist of processing power, virtual machine, and Storage, etc.
- 2- *Broad network access* Talk about properties beforehand could be achieved by using different system Gadgets, like a laptop or else mobile phones.
- 3- *Resource pooling* Cloud management vendor pool; their properties were subsequently numerous clients. This is said to be multi-stage; for instance, the physical server might have some virtual machines that have a place with unique customers(Zhang, Cheng, and Boutaba 2010).
- 4- *Rapid elasticity* Customers could rapidly get more properties. It can be deleted from the cloud through expansion and can be expanded by freeing these properties when they were not at all extended needed.
- 5- *Measured service* Resource utilization monitor usage of storage, CPU time, usage of bandwidth, and many more. The top of metrics applied to completely clouds unless

each one of cloud delivers users with different levels of service Abstract, it is an alternative method of management (Patel et al., 2013).

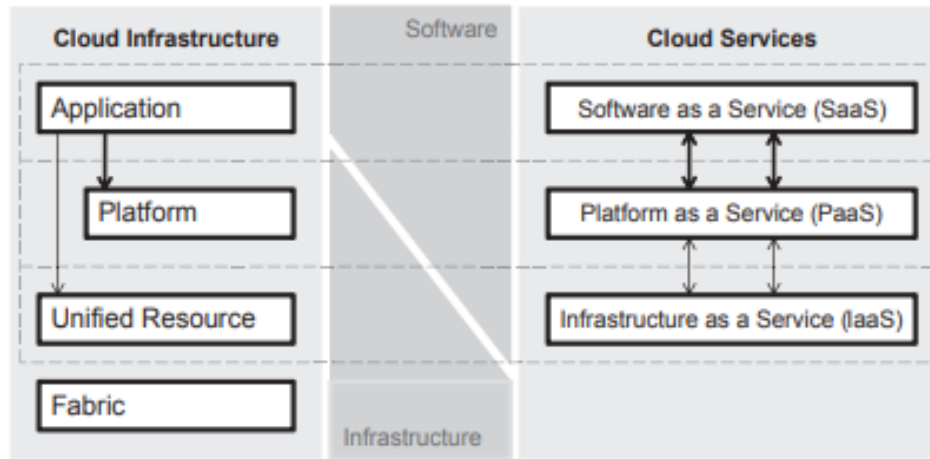


Figure 2. 1 Cloud Architecture regarding Cloud services (Patel *et al.*, 2013)

2.2.4. Cloud Deployment Models

NIST had a recognized four cloud computing standard models that can be applied to meet the changing requests of operators or suppliers. These models: private, community, public, and hybrid models fluctuate in terms of whereas the hardware was located, the unit of responsible aimed to maintain the system, besides who could use system resources (Ganapathi *et al.*, 2010). There were three main deployments of cloud models shown in (Figure 2.2).

1. Private cloud: a property of a single organization or individual. Also, the operability cloud allows a proprietor or a third gathering to use it.
2. Public cloud: Clouds owned by large cooperatives are available for public use and require investment.
3. Community cloud: the cloud that was mutual via multiple organizations and had attributes that meet al l necessities.
4. Hybrid cloud: the cooperation of 3 clouds assumed previously. The cloud can be accomplished separately; nevertheless, data then applications go over the hybrid cloud. Emergencies can similarly occur in hybrid clouds, which may make private clouds public.

2.2.5. Service models of cloud computing

computing of cloud deliver facilities through 3 models, platforms and software, infrastructure (Hwang *et al.*, 2012).

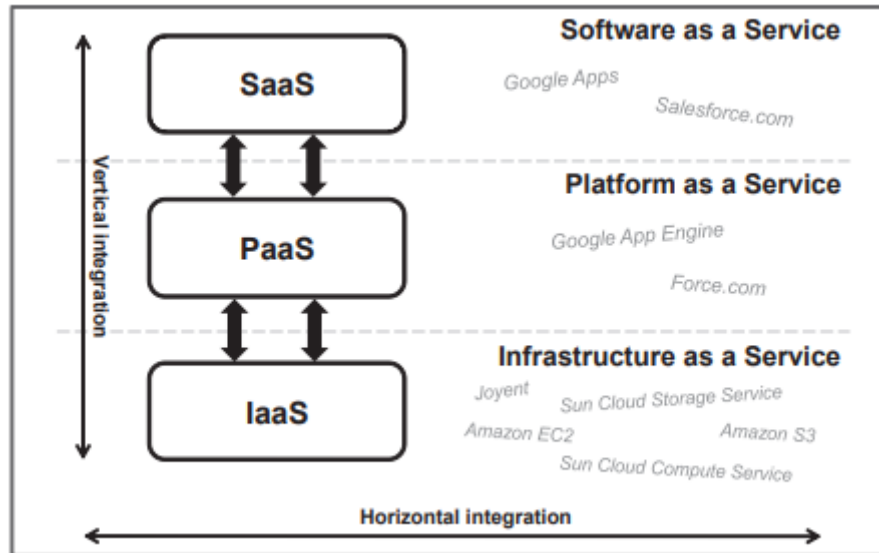


Figure 2. 2 Cloud Architecture (Patel et al. 2012)

1. Software as a Service (SaaS): Delivery of model through software that requests can be recovered through modest boundaries (example: browser on the Internet). Users don't care about the hidden cloud infrastructure, or even the network, operating system, server, storage and point. That model furthermore eradicates the requirement to set up and operate applications on the domestic computer. Applicable to WebMail, Microsoft Online, NetSuite, Google Docs, Facebook, MMOG Games, and other samples (Cusumano, 2010).
2. Platform as a Services (PaaS): Paas delivered an advanced composite setting to form, test, organize and host applications produced or learned by customers. In general, designers may receive specific limitations for the kind of software which could be written in interchange intended for built-in scalability application. PaaS clients cannot be organized like SaaS clients, then can governor the deployed applications and their hosting setting configuration. Several samples of PaaS were Engine Yard, Windows Azure, Google App Engine, Force.com, Heroku and MTurk (Boniface et al . 2010).

3. Infrastructure as a Service (IaaS): It was delivered for users with treating, networking, storage, and so on essential computing resources. IaaS clients could organize and use any application, software, and operating system. They dynamically expand and shrink. Samples of IaaS contain Amazon EC2, VPC, Eucalyptus, Flexi Scale and Rackspace, IBM Blue Cloud (Demchenko et al., 2011).

2.2.6. Problems in cloud computing

1. Determinants of privacy and security: The biggest and significant disadvantage of cloud computing is the problem of user identity and data security (Venters et al., 2012). Data be appropriate for the organization to have been used cloud services drive to store in a public setting. Compared with the non-shared setting, the security of the shared setting is implicitly insecure. Also, the storage and distribution of delegated data were not gone for the organization of the legal and regulatory obligations surrounding the facts. Serious problems include the security model of vulnerable vendors, customers' inability to answer for review results, responsibilities for indirect administrator, and proprietary implementations, all of which could not be inspected and cannot control physics (Lu et al . 2013).
1. Closely connected for those subjects related to rule and data supply are the guard for data plus other possible security vulnerabilities, which are caused by resources being shared among multiple tenants and where the resources may be unidentified. In specific, sensory data or else guard applications were pivotal to outsourcing subjects. Although the data shall be guarded in the formula of legal subjects regarding the location of the data, it shall still be accomplished by the system. Due to numerous applications for the cloud, the system plus various cloud kinds mean that the user's security model and requirements are different (Mezgár and Rauschecker, 2014).

2. The determined Data storage: The main issues in the data storage are data loading taking on consider isolation management / multi-tenancy, storage controllers, particular failures, and data exposure to third parties (Zhang et al., 2010).
3. Interoperability and Standardization: The ability of systems and services that create, exchange and consume data plus standardization have been a massive influence on cloud acceptance and usage. Uniformity would be improved and quicken the acceptance for cloud computing because operators would be had more choices in the cloud except for vendor lock-in, movability, plus the capability to using cloud services providing by many customers. That contains the ability toward the use of the organization's identifiable existing data center references, ideally. "“ The biggest challenge facing long-term adoption of cloud computing services is not security, but cloud interoperability and data portability ”" IEEE cloud computing experts (Venters et al., 2012) “The nonexistence of combination between these networks varieties it difficult for organizations to integrate their IT systems into the cloud and achieve productivity improvements.

2.3. Scheduling in Cloud Computing

Cloud Computing's task scheduling targets provide optimal tasks for users and simultaneously deliver the maximum cloud system performance and QoS. Broad objectives include load balancing, service quality, economic theory and maximum system output and time(S. Kaur et al. 2019).

Resource scheduling in cloud computing is a challenging job, and the scheduling of appropriate resources to cloud workloads depends on the QoS requirements of cloud applications (Singh et al., 2016). The schedule was used within cloud computing can accomplish high performance with optimal system throughput. Efficiency, speed besides resource utilization in a heightened manner principally depend on the scheduling category of the cloud computing setting. The various standards for scheduling were maximum CPU utilization and maximum outcomes (Lakhani *et al.*, 2013).

For its business model, cloud computing was disappointing. Operators, mixed approaches and working conditions also have issues with various workers. The distribution and spread of social income and prosperity was identical in the cloud computing system: the references

given by builders of infrastructures were equal to all social prosperity. Various operators need different methods of tasks that can be represented as group personalities. The amount of references paid by users could be observed when paying for the social people via the application. They distribute unlike wealth according to differences (Tiwari, 2014).

Novel Berger model of equitable scattering of cloud and effectiveness (period and cost) that is generally founded by subsequent points (Lakhani *et al.*, 2013):

1. Cloud computing provides multiple users with an open application and resource server.
2. Task scheduling algorithms are typically based on cost or performance. There were different task, including minimum delivery time, maximum availability and lowest cost. This had exact objectives. Unnecessary task scheduling merely meets the balance between performance and resource, but also the equal assignment of references.
3. QoS is based on the interest of a customer, expenditure criteria otherwise and at the time finds the best benefit or balance. Finally, a win-win situation for consumer performance and cost efficiency will be achieved
4. Using virtualization technology to package and provide resources to users. These novel features necessitate us to create a link among operators besides virtual references. Furthermore (Raj *et al.*, 2013)s, we prerequisite to advance new-fangled appropriate job scheduling and reference mapping devices.
5. Improving the level of Service of quality is to enhance customer satisfaction; the main technical fairness and satisfaction strategy. Therefore, the benefit comes first, then reflect both equality and cost (Jula *et al.*, 2014).

2.3.1. The Environment Features of Job Scheduling within Cloud Computing

Cloud computing setting, job scheduling and reference allocation succeeded by the provider over virtualization technology. Patel *et al* (2013) were used it to flog and inclusive operator jobs transparently. A job scheduling strategy that focuses only on fairness or effectiveness will raise the cost of period, space, and throughput, and at the same time, advance the service of quality of whole cloud computing. The features of job scheduling within a cloud computing setting were as shadows (Patel *et al.*, 2013):

1. Mission planning serves a unified resource platform: When using virtualization technology for cloud computing, we highlighted physical references (wholly kinds of hosts, workstations, even PCs ...) into a united reference pool and shielded mixed references to provide higher utilization rates.
2. Task scheduling is globally centralized: Provides central resources to multiple distributed applications through mirroring services, virtualization technology, and mirroring services that enable cloud computing job scheduling to achieve global centralized scheduling.
3. Every node in the independent in cloud: Each node of cloud stands as separate, in addition to scheduling program within the cloud does not intervention thru a strategy of the scheduling of identical nodes.
4. Task scheduling scalability: The cloud provider's resource supply scale may be incomplete. The size of resources can become more important as the application requirements continue to increase, and the various computing resources increase (Manvi and Shyam, 2014)
5. Task scheduling is dynamically adaptive: Depending on the needs, applications may need to be scaled up and down in the cloud. Virtual computing resources in the cloud system can expand or shrink at the same time.
6. Task scheduling plan set: Divide task scheduling into two parts: the first is the main one responsible for arithmetic programming interfaces (APIs) and application scheduling, and this is used as a unified resource pool scheduler, while the other is task scheduling. The other is for unified port resource scheduling in the cloud. Each schedule includes two bidirectional processes: the scheduler leases resources from the cloud, and the scheduler calls back the requested resources after usage (Černý et al . 2017).

2.3.2. Task Scheduling goals in Cloud Environment

Cloud computing task scheduling intends to afford users with the best task scheduling and at the same time, provide throughput and service of quality for the entire cloud system. Particular aims were loaded balancing, quality of service, optimum runtime and system throughput (Lakhani *et al.*, 2013).

1. Load balance: Load balancing and task scheduling are closely related to each other in a cloud environment, and the task scheduling mechanism is responsible for the optimal matching of tasks and resources. Due to the applicability of task scheduling algorithms, load balancing has become another important measure in the cloud (Lakhani et al., 2013).
2. Quality of Service: Cloud computing provides operators by needs, such as services and storage resources, which are produced and implemented within a type of service quality. Whereas job managing of scheduling involved task allocation, the service of quality of resources must be guaranteed (Jelassi et al., 2017).

Multiple applications that help accelerate the Internet of Things (IoT) wherever the cloud is. There are still some fundamental problems, such as: unable to understand the poor confidentiality of location services, lack of mobile support, unnecessary network consumption of bandwidth, uncertain third-party security topics, and other reasons, all refused in particular communications and real-time environments The reason for using it. Application to improve service quality (Bedi *et al.*, 2018).

Cisco's fog adopted for computing to resolve the quality issue. Fog computing is nodes that work between devices and the cloud and can be allocated throughout the network. They found that fog computing works efficiently providing quality of service in the cloud environment (Munir et al., 2017). Their contribution was ensured better performance and more efficient in processing requests of cloud computing services. Algorithms that assist in executing services are placed in virtual machines that are included in the cloud computing structure. There were various algorithm approaches to advance service of quality within cloud computing, yet they vary within their execution time of service (Munir *et al.*, 2017).

3. Best running time: Tasks mainly used for services can be divided into different categories according to user needs, and then set the optimal running time and priority of different goals for each task. It will indirectly improve the service of quality of task scheduling in the cloud environment.
4. The throughput of the system: Throughput is essentially an indicator used in cloud computing to measure cloud task scheduling optimization and target performance. This is an indicator that must be considered in business model development. Increasing the throughput of users and cloud providers will benefit both of them.

2.3.3. Scheduling Strategies

Job scheduling is used to allocate certain jobs to specific resources at specific times. In cloud computing, problematic job scheduling is a major and exciting problem. Therefore, the work planning process should be active. An effective job planning strategy must aim at reducing the time of response; consequently, the submitted work can be completed in the shortest time imaginable, and there will be a surge in revenue within a specific time. Therefore, customers can submit fewer job receipt rejection positions and more work to the cloud, which ultimately shows the cumulative consequences of accelerating cloud business concerts. According to different standards, there were different categories of scheduling, such as static and dynamic defined below, centralized and distributed, offline and online, etc (Singh et al., 2016):

1. Static Scheduling: The pre-planning tasks can identify all the information about available resources and tasks, as well as the functions assigned to the reserve fund at a time, so it is easier to adjust according to the planner's prospects.
2. Dynamic Scheduling: There are jobs dynamically, which are used to schedule through the scheduler over time. It was the most elasticity in static scheduling and can run decisively. In order to acquire a stable, accurate and useful scheduler approach, it becomes more serious about taking load balancing as the primary factor.
3. Centralized Scheduling: described previously in the scheduling of dynamic, it is the responsibility of the central / scheduler distributed to create selections globally. Key benefits of centralized scheduling were comfortable to work; effectiveness, more controlling and care resources. In contrast, this scheduler absences scalability, accountability besides effective performance. Because of this shortcoming, it is not cumbersome for large grids.

4. **Distributed / Decentralized Scheduling:** Although its capability is weak, it is more realistic for the actual cloud for unified scheduling. Since there is no critical control entity, the local dispatcher is required to reach and maintain the status of the job queue.
5. **Preemptive Scheduling:** All jobs toward being inserted through implementation; in addition, one job could be passed to different references for the resource it owed initially, which can be used for other jobs. This type will be more helpful if you handle restrictions such as priorities carefully.

2.3.4. Scheduling Process

The process of scheduling in the cloud could be divided into three stages, references identification besides screening – Datacenter Broker determined references existing in the network solution and gathers state information interrelated to them. Reference choices-select board resources based on the determined parameters of tasks too resources. This is a crucial phase. Task submission-the job submits toward the specified source.

2.3.5. Scheduling Criteria

As mentioned above, various CPU scheduling algorithms in this field have different attributes. The choice of a particular algorithm may make one type of process better than another. In order to select an algorithm for a specific situation, we must consider the properties of various algorithms. Many standards have been proposed to compare CPU scheduling algorithms. Whose characteristics are used for comparison and which characteristics can make a significant difference in the algorithm to determine which is best? The conditions include the following details (Lakhani *et al.*, 2013):

1. **Context Switch:** This is the method of storing and scheduling the context (state) of the preempted process so that implementation can be improved from the same point later. It is usually computationally concentrated, resulting in wasted period and storage, which converted to rises the scheduler mentioned previously. Hence, the design of the operating system is to optimize these switches only, intending to minimize them (Singh *et al.*, 2016).
2. **Throughput:** be present cleared as the numeral of procedures finalized each unit time.
3. **CPU Utilization:** it's a measurement of how much full of activity of CPU is.

4. **Turnaround Time:** approximately the overall period it takes to complete the process, and the time it takes to execute the process. The time interval from the submission process to attainment is turnaround time. The total turnaround time is the sum of time spent waiting to enter memory, waiting time in the ready queue, execution time on the CPU, and execution I / O.
5. **Waiting Time:** The overall period of a procedure had been waiting in the ready queue. The CPU scheduling approach affects only the quantity of time that a process spends waiting in the ready queue.
6. **Response time:** This is the period as of submitting a request to generating the first reply. Consequently, time responded must be short of achieving the best scheduling. Therefore, we can implement an intelligent real-time shared system scheduling algorithm must have the following purposes: minimum context switching, maximum CPU utilization, maximum throughput, minimum turnaround time, minimum waiting time. Because these scheduling approaches have many shortcomings, they were widely used to expect scheduling of round-robin in time-sharing as well as real-time operating systems. Also is reflected to be the greatest commonly applied CPU scheduling algorithm.

2.4. Existing Scheduling Algorithm

The scheduling approach is preemptive and non-preemptive. Numerous scheduling approaches at present exist within the cloud, and examiners focus on established scheduling of task approaches that would be argued in the present sector.

1. **First Come First Serve Algorithm:** all jobs in the queue starts with assistance. The algorithm is fast and straightforward (Frijns et al . 2014)
2. **Round Robin (RR):** RR is a preventative direct scheduling algorithm used to mass-sell tasks in an already-used framework in which task execution is still performed after a certain period called time-critical (Rajput et al., 2013)

The main characteristics of RR are:

1. The use time interval is too short will cause too many context switches and reduce CPU efficiency.

2. The quantum reason for the long usage time; within particular cases, time responded is also reduced close to FCFS performance. Figure (2.3) the main steps of applying RR (Yassein *et al.*, 2013).

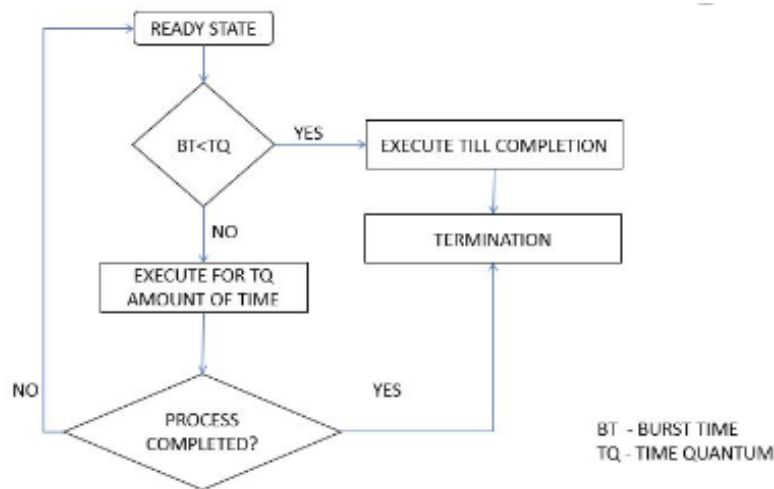


Figure 2. 3 Original Round Robin Algorithm (Yassein *et al.*, 2013)

Round Robin is given by the following steps:

1. The scheduler maintains a queue of ready processes and a list of methods that have been blocked and swapped out.
2. The process control block of the newly created process is added to the end of the ready queue. Process control Delete the termination process block from the scheduling data structure.
3. The scheduler always selects the process control block from the beginning of the ready queue. This is a disadvantage that all processes have the same priority. Round robin also facilitates this process CPU bursts are short and penalize longer CPUs.
4. When the running process completes its time slice, it will move to the end of the ready queue. Each cycle, algorithm spends time on the processor each cycle. The process is executed on a first-come-first-served basis, but it is preempted after a time slice. Process Will be completed in the given time slice. Otherwise, the process will return to the end of the ready queue and Return to the processor later.

5. Input and output operations are waiting for are completed, or the process is swapped in its process. The control block will be moved from the block/swap list to the end of the ready queue.

Numerous researchers have suggested different variants of the RR approaches (Lin *et al* 2011). In (2016), a method termed dynamic round robin (DRR) was intended for scheduling and integration of energy-aware virtual machines. Related to the schedule of GREEDY, RR, and power reduction. DRR shows advantages in decreasing power consumption (Elmougy *et al.*, 2017) Improvement of the traditional RR by a random cycle of RR depends on selected round randomly, used in progressions from dissimilar operators to accomplish the best choice of work to be served. Simulation utilizing cloudsim simulator V 3.0 to investigate the performance of suggested solution based on different evaluation indicators (similar an average of throughput besides an average of turnaround time) (Raj *et al.*, 2013).

2.5. Shortest-Job-First (SJF)

SJF scheduling allocates jobs rendering to the shortest period of a burst. It can be preventative or non-preventative, and the non-preemptive SJF approach would permit the presently running procedure to complete its burst of CPU. Preemptive SJF scheduling is now and then referred to as the shortest retention time first (SRTF). While it provides the best average time of waiting, it is more appropriate for batch processing frameworks, but will suffer from starvation (Lakhani *et al* . 2020).

This algorithm considered for maximizing the throughput. This hint exemplified as appears in Figure (2.4).

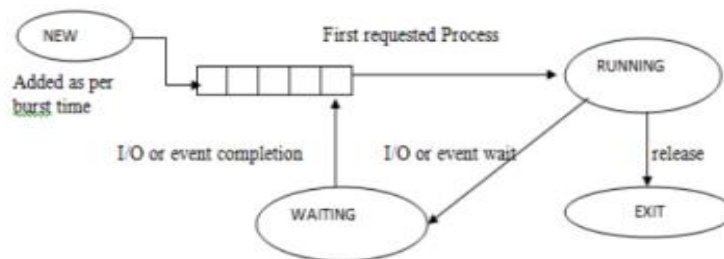


Figure 2. 4. SJF algorithm (Alworafi et al., 2017)

The major characteristics of SJF are:

1. SJF algorithm real concern on to determine the size of subsequent CPU call.

2. SJF reduces the average of the waiting time as it assists tinny procedures before serving massive procedures.

Numerous academics research investigated different variants of SJF. For example, Jia and Keung suggested a scheduling algorithm combined with job grouping, priority awareness SJF (shortest job first) in 2013 to minimize the waiting time, shorten span, and maximize resource utilization.

The suggested scheduling algorithm purposes of decreasing processing time, waiting time and overhead. In the trial, a Gaussian distribution was used to generate tasks, and a random distribution was used to create resources. The CloudSim framework was used to simulate the projected procedure under numerous conditions (Zhang et al., 2015). Compared with the traditional RR SJF (RRSJF), it first selects the process based on the shortest job cyclically to provide the best job selection. A simulation conducted by a CloudSim simulator V 3.0 to investigation the performance of the suggested solution based on different evaluation indicators (such as average turnaround time, average wait time and context switching).

2.6. Related Works

Cloud computing was yet a new field for study, so far, not much had been made in this area. Regarding the articles and books that reviewed some detailed knowledge and improved the scheduling approaches that have been developed so far. In recent years, the development of this field is very rapid, so the amount of ongoing research is large.

A new scheduling algorithm, called a meta-scheduling algorithm, is introduced. They divided their work into four different categories, namely short and narrow, short and wide, long and narrow and long and wide. With this division, they can assign more priority to shorter jobs without having to sacrifice the lengthiest time ended of larger jobs. While we look at it differently, this is based on the standard of slag time. They will delay the lengthiest work to the greatest point and formerly interruption their processing consequently. It is essential to avoid starvation; this is the main goal of the algorithm, so starvation will not delay the work beyond the slag discharge (Hausmans *et al* . 2013).

By complexity attention to choose algorithm, which is simple or easy to use in processing, FCFS Algorithm is the simplest Scheduling Algorithm. Both SJF and RR Algorithm depends heavily on the size of time quantum and become difficult to understand and code, excluding

the performance and optimal scheduling. Another method of cyclic CPU scheduling algorithm improved, which can enhance the execution power of the CPU continuous working framework.(Rajput *et al.*, 2013)

Round Robin Algorithm has the largest waiting time whereas the priority algorithm has the least waiting time than FCFS Algorithm and SJF Algorithm In case of FCFS Algorithm, CPU is allocated in the order in which the processes arrive. In contrast, in SJF Algorithm CPU is allocated to the process with least CPU burst time (Kaur *et al.*, 2014).

2.7. Similar Approaches

Ravel and Elhajj recommended the priority scheduled starvation Avoidance Plan (SAF-PS), which uses the time to keep packages to alleviate starvation in addition to diminish the drop rate. This strategy helps achieve non-linear timing that may increase overall productivity and requires their algorithmic support. Also, single support can be locked in the following ways to increase the loss rate: Secondly, the updated data packets can be updated regularly to drop them when the packet flow is dropped due to insufficient space in a certain queue. The better expectable planning maybe to renew the package with less upper activity (Jabbour and Elhajj 2008).

A novel $v(t)$ controlled CSMA approach is proposed, that could be implemented in an appropriate manner using RTS / CTS instruments. Perform linked scheduling to support connections with lengthier queues to reduce normal delays (Xue *et al.*, 2012).

Round-Robin CPU Scheduling approaches depend on round-robin algorithms. Because RR coordinates the benefits of the demand schedule. The proposed algorithm implements many ideas by allocating new requirements to the process. The current RR CPU scheduling approaches do not attain a continuous work-frame due to the high switching rate, long holding time, long reaction time, long turnaround time and low throughput. The implementation of the time-imparting framework could be improved to the suggested approach and can also be changed to progress the ongoing framework execution. The suggested approach boosts the disadvantages of RR-CPU scheduling approach (Yassein *et al.*, 2013) altogether.

The proposed architecture of a scheduling system for a cloud storage setting takes into account multiple conditions and shows the versatility of scheduling programs in the cloud.

The main purpose of this article is to maintenance large-capacity storage of streaming media within cloud storage and afford continued access to the required data to its finishing operators when virtual machines were migrated (Wu *et al.*, 2013).

Aggarwal and Nagpal have established a unique method for the scheduling SJF of approach, which helps reduce starvation problem within a ruggedly stacked PC framework. The ASJF approach reduces the starvation problem in basic SJF structural planning. The ASJF approach is based on SJF and multi-level annotation queue scheduling (MFQS) technique. Besides, SJF and ASJF were inspected at close range to show the comparison between them. The ASJF algorithm proves the maximum excessive CPU usage and effective processing of properties. Since the MFQS approach is used in conjunction with SJF, the process of separating processes into different queues then exchanging them between them will advance reduce the starvation problem (Aggarwal *et al.*, 2016).

2.8 Tools

In (Buyya *et al.*, 2009) research on the use of genetic algorithms to deal with cloud scheduling problems, we proposed PGA to achieve cloud scheduling problem optimization or sub-optimization. Mathematically, we treat the scheduling problem as an unbalanced distribution problem.

It has been discussed that by monitoring the original parameters of the virtual machine in real-time, once these parameters exceed the threshold, it is easy to detect overload. The ant colony algorithm can quickly find the adjacent idle nodes from the resources and start the virtual machine to bear part of the load and meet these representations and resource requirements of the load. In this way, adaptive dynamic resource scheduling of goods is realized on the cloud service platform, and the purpose of load balancing is achieved. (Jeyarani *et al.*, 2010)

Huang Lu and Chen Haishan proposed a solution architecture that satisfies user resource requirements cost-effectively and discussed scheduling schemes that use heterogeneous clusters while providing good performance and fairness (Liu *et al.*, 2014). Development share is used as a shared indicator. By looking at various possible configurations in a heterogeneous environment, we can reduce the value of maintaining such clusters by twenty-eight per cent.

The proposed scheduling approach simultaneously provides excellent performance and fairness in a changing cluster. By using progress share as a shared indicator, we can increase the performance of CPU-enabled jobs by 30% while ensuring fairness among multiple jobs.

As related work illustrated that have been different types of job scheduling algorithm applied in the cloud environment with suitable modification. The main aim of any job-scheduling algorithm is to maintain fairness among the jobs for their execution and reduce waiting time and to improve performance, quality of service like throughput end-to-end delay. In this Thesis we tried to propose a novel hybrid algorithm which is leading to achieve a balance between response time and waiting time, reduce gap between them. In addition to, we had been used a modified round robin not static and short job first that have a dynamic quantum time for each iteration to reach as much as a lowest waiting, and turnaround time.

There are various issues with algorithm scheduling based on specific optimization parameters. The two criteria needed in batch systems are time and efficiency, response time and fairness are the two criteria needed in the interactive system, while in the real time system the time limits are relevant. A scheduling algorithm must therefore be chosen so as to fulfill the requirements needed and provide successful service and resource allocation.

2.9. Conclusion

The biggest challenge in the scheduling model is the starvation problem. Among them, the execution of tasks cannot be completed at all. Starvation is a problem usually faced in multitasking where a process is continuously denied necessary resources. The proposed method focus on dealing on this problem based on hybrid two most common scheduling algorithms to avoid disadvantages of each other so that the evaluation of the performance metric increases especially waiting time, response time, throughput rather than decrease and reducing probability of starvation occurrence to be zero at possible as. It is usually caused by an error in the scheduling algorithm, not a resource leak, and maybe intentionally caused by a denial of service attack (such as a fork bomb). Starvation is a problem often encountered in multitasking. In this task, a process is uninterruptedly denied the references it needs.

The suggested method focuses on solving the starvation problem based on a hybrid of the two most common scheduling algorithms, to avoiding each other's disadvantages, so the

assessment of performance indicators especially rises the waiting time, response time, and throughput slightly reduced and reduced the possibility of starvation.

CHAPTER THREE: HARSQ METHODOLOGY

3.1 Overview

This chapter focus on the practical part of the proposed approach and describe the process of the all component of it, starting with task data arrange it in the queue then applying dynamic Round Robin algorithm, throw all method for achieving the main objective of the thesis.

3.2. A hybrid approach (HARSQ)

This thesis aims to address starvation problems using a new hybrid scheduling technique that relies on scheduled algorithms called HARSQ, such as SJF and RR. The HARSQ reduces the chances of starvation to the maximum extent possible by increasing performance metrics and performance efficiency by avoiding weaknesses from SJF and RR. Avoid high traditional wait times and non-recurring deadlines that are encountered by cutting off time and portion; this is performed by the RR algorithms in HARSQ. The RR quantitative of process time is very interesting because the quantitative time as concise as many of the keys to context exchange reduce the CPU efficiency while setting the quantity for a long time may lead to weak response time and approaching FCFS algorithm (First-Come-First-Serve) (Rajput *et al.*, 2013; Tiwari *et al.*, 2014).

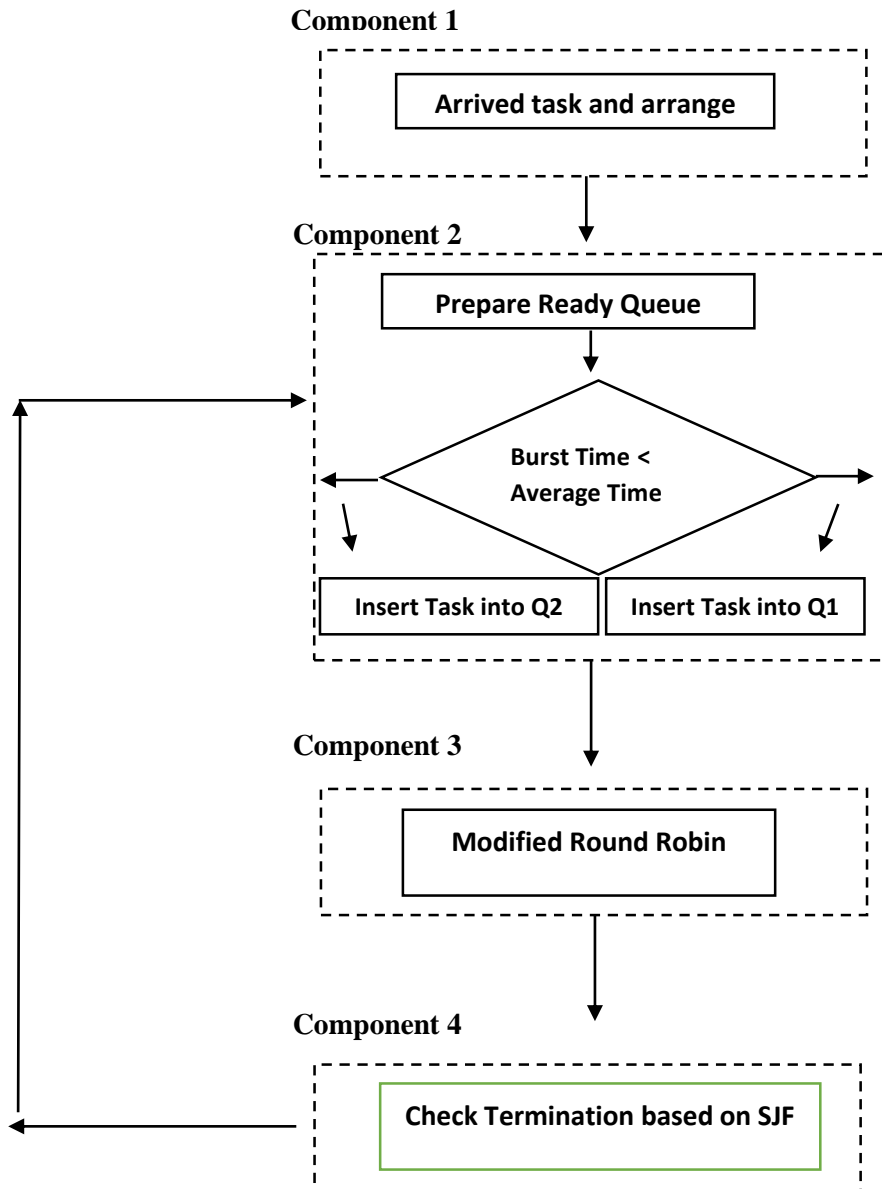


Figure 3. 1 HARSQ algorithm Flowchart

HARSQ is designed as a component-based algorithm taking a Queue data structure type for optimizing the execution time as possible as achieve a potential. HARSQ involves four main components, as follows:

Component 1: Hold on and Arrange all submitted tasks, $T_i, I=1, 2, \dots$, several submitted tasks, according to their burst time, and arrived time if they have the same burst time. **Input:** random arrived task. **Output:** arranged task based on burst time and arrived time Process:

$$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(B_{ij} + (-1)^{1-\alpha} \cdot q_{i(j-1)})^2} \dots\dots\dots \text{Eq 1 (Nayak, Kumar Malla, and Debadarshini 2012)}$$

Component 2: the configuration of the queue data structure depends on

- Calculate the average burst times for all tasks reached
- Depending on the task's burst time, divide each task (Queue1 for the task with burst time less than or equal to the average, otherwise insert T into $Q2$).

Component 3: apply modified RR by the following step

- Depending on the task source queue that is currently performed, a quantum of (q_{ij}) is calculated (whether it is from $Q1$ or $Q2$), and the round to be executed in, as follow

Where q_{ij} is the quantum at iteration j , $i:1, 2, \dots, n$ and, B_{ij} is burst time of task i at iteration j , $q_{i(j-1)}$, and α is a binary selector $\alpha = \{0,1\}$ regarding to queue number.

Equation1 adjusts the time quantum according to the burst time of processes founded in the ready queue. Initial time quantum could be calculated by the median of burst times for the set of processes in ready queue then it is change dynamically to each task over each round based on adaptive

Component 4: apply modified SJF by the following step and check termination

- Allocate the first two tasks of $Q1$ to the resources followed by the primary task of $Q2$.
- Until emptying $Q1$ and $Q2$, the final steps are continuously repeated.

Below is a HARSQ pseudo-code

"Input: Tasks

Output: Rescheduling all tasks with balance

Define Variable

T_i: Task *i*

B_i: Burst time of task *i*

RQ: Ready Queue *Q1*, *Q2*: $Q1 \cup Q2 = \text{Ready Queue}$, $Q1 \cap Q2 = \varnothing$ select counter to indicate whether the selected task is from *Q1* or *Q2*

count_iteration: an initialize value for iteration *j* and quantum (*q_{ij}*)

q~: average of burst time of tasks.

q_{ij} : quantum time assigned to task, *T_i*, in the iteration *j*

Select (): a function to select a task, *T_i*, from *Q1* or *Q2*

Execute (): a function executes a task, *T_i*

BEGIN

Arrange arrived tasks in Queue burst time and arrive time SJF

q~ = the average of burst time of all tasks Divide *RQ* into *Q1* and *Q2*

For each task *T* in *RQ*

BEGIN

IF *B(T)* > *q~* **THEN**

 Insert *T* into *Q1*

ELSE

 Insert *T* into *Q2*

END

Start iteration

Task selector = 0

WHILE (*Q1* & *Q2*) are not empty

BEGIN Apply RR algorithm to select the first two tasks from *Q1* and the first task from *Q2*

IF Task selector > 2 **THEN**

BEGIN

T_i = Select(*T* [Head (*Q1*)]) Execute(*T_i*)

 //Determine the new quantum time

IF *count_iteration* = 0

THEN *j* = 1 and *q_{i(j-1)}* = 0 *q_{ij}* = *q~* + *q~* (*B_{ij}* - *q_{i(j-1)}*)/2

select++ *count_iteration* ++

END

ELSE

BEGIN

T_i = Select (*T* [Head (*Q2*)])

 Execute (*T_i*)

q_{ij} = *q~* + *q~* (*B_{ij}* + *q_{i(j-1)}*)/2

END // Rem *B* terminated is the finished task burst time

IF *B(T)* = 0 **THEN** *q~* = *q~* - *q~* *B* terminated

IF new task (new *T*) is arrived **THEN**

BEGIN

IF *B*(new *T*) > *q~* **THEN**

 Insert new *T* into *Q1*

ELSE Insert *T_{new}* into *Q2*

q~ = *q~* + *q~* *B_{new}*

END"

3.2.1. Component 1: Arrived task and arrange them

To apply the proposed hybrid by selecting eight tasks and arrange them to (Q1 and Q2) as shown below in Table 3.1

Table 3. 1: Arrange task based on burst time and arrive time (SJF)

Task (T)	Burst time	Arrival time	Arrange task based on burst time and arrive time	Ready Queue
T1	35ms	0		T3
T2	12ms	0		T4
T3	8ms	0		T6
T4	10ms	0		T2
T5	23ms	1		T7
T6	10ms	1		T5
T7	19ms	2		T8
T8	30ms	3		T1

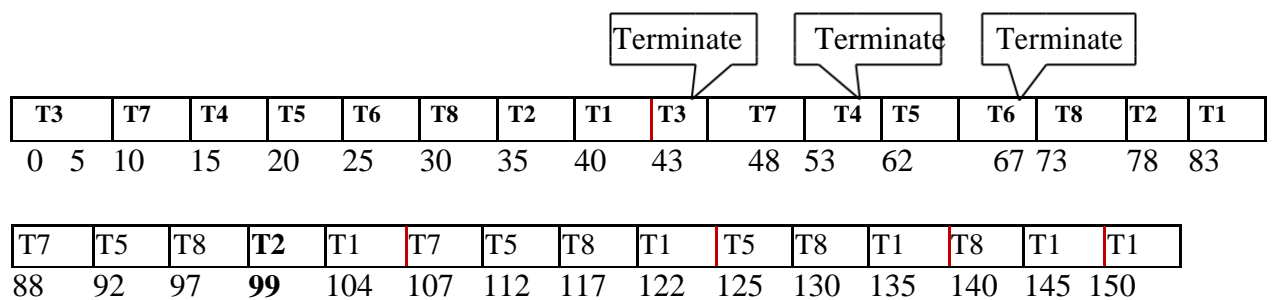
3.2.2. Component 2: Prepare a Queues data structure

In this component, we use the SJF technique as the following table

Task3	Task4	Task6	Task2	Task7	Task5	Task8	Task1
Q1				Q2			

Task 3	Task 4	Task 6	Task 2	divided depend on average = 18.30		Task 7	Task 5	Task 8	Task 1
-----------	-----------	-----------	-----------	--------------------------------------	--	-----------	-----------	-----------	-----------

In SJF and RR Time quantum = 5



3.2.3. Component 3: Modified Round Robin Algorithms

Table 3. 2: A Result Case Study

Responding time (ms)				Waiting time (ms)			Turnaround time (ms)		
T	HARSQ	SJF	RR	HARSQ	SJF	RR	HARSQ	SJF	RR
Task1	35	101	0	115	101	111	150	136	146
Task2	30	28	5	87	28	73	99	40	85
Task3	0	0	10	37	0	45	43	8	53
Task4	10	8	15	43	8	48	53	18	58
Task5	15	58	20	101	58	102	124	70	125
Task6	20	18	25	56	18	57	66	27	67
Task7	5	40	30	77	40	93	105	56	111
Task8	25	71	35	107	71	118	137	98	138
Average time	17.5	40.5	17.5	77.875	40.5	80.875	97.125	52.87	97.875

One of the most frequently used self-modification algorithms is Round Robin despite its design problem due to a predetermined quantum of time (Yassein *et al.*, 2013). Component three of the hybrid approach modified RR algorithm to include it as a necessary process in the proposed method. The main problem with RR scheduling algorithm is that the time quantum is usually fixed in the traditional RR and does not change during the process between 10 to 100 milliseconds (Lakhani *et al.*, 2020).

The performance of the RR algorithm depends on the size of the time quantum. The context changes more with the smaller time quantum than the larger quantum of time, so that response time is more. Choosing a suitable quantum of time is very necessary because the overall performance of the regulations may be reduced with the quantitative choice of weak time. This thesis presents a new approach to solve the persistent starvation problem.

3.2.4. Component 4: apply SJF and check termination

As indicated from the previous table (3.1), there is an aggressive or modest response to the proposed technique, and the response and waiting time are consistent with the traditional RR. It was also noticed that there is a great value for the proposed procedure in terms of response and waiting time compared to SJF, although this has a better response time. Chapter 4 includes fixing these problems by applying more experiments to achieve the thesis objectively.

3.3. Manual Case study based on HARSQ

We apply a manual test time quantum determined by a proposed HARSQ we used proposed Approach by choosing two tasks, one from Q1 and another from Q2 alternatively as follows:

Table 3. 3: Component 1 Arrived tasks

Tasks	Task1	Task2	Task3	Task4	Task5	Task6
Burst time	12s	8s	23s	10s	30s	15s
Arrival time	0s	0s	1s	2s	3s	4s

Table 3. 4: Component 1 Arrange Tasks According to SJF

Task	Burst time	Arrival time	Arrange task based on burst time and arrive time	Ready Queue
Task1	12ms	0ms		Task 2
Task2	8ms	0ms		Task 4
Task3	23ms	1ms		Task 1
Task4	10ms	2ms		Task 6
Task5	30ms	3ms		Task 3
Task6	15ms	4ms		Task 5

Table 3. 5: Component 2 Divide Queue for two queues

Average of burst time			$q = (12+8+23+10+30+15)/6 = 16.3$		
Queue 1			Queue 2		
Task2	Task4	Task1	Task6	Task3	Task5

Table 3. 6: Component3 Apply RR (quantum calculations round1)

Task	$B_{ij} = B_i(j-1) - q_i(j-1)$	$(B_{ij} + q_i(j-1))^2$	Queue Number	$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(B_{ij} + (-1)^{1-\alpha} \cdot q_{i(j-1)})^{\alpha}}$
Task2	8-0= 8	$(8+0)^2 = 64$	Q1	q21= 16.71
Task4	10-0= 10	$(10+0)^2 = 100$	Q1	q41=16.63
Task6	15-0= 15	$(15-0)^2 = 225$	Q2	q61=16.56
Task1	12-0= 12	$(12+0)^2 = 144$	Q1	q11=16.59
Task3	23-0= 23	$(23-0)^2 = 529$	Q2	q31=16.52
Task5	30-0= 30	$(30-0)^2 = 900$	Q2	q51=16.51

Time-based on the table (3.6)					
Task2	Task4	Task6	Task1	Task3	Task5
8s	18s	31.56s	43.56s	57.08	70.59s

Quantum calculations round2

Task2, Task4 and Task1 are all finished in the first round so q^{\sim} will be updated as

- After ruining Task2 , B terminated = 8 , $q^{\sim} = 16.3 - (13.5/8) = 14.61$
- After ruining Task4, B terminated = 10 , $q^{\sim} = 14.61 - (11.81/10) = 13.42$
- After ruining Task1, B terminated = 12 , $q^{\sim} = 14.42 - (10.62/12) = 12.52$

Equally, three tasks running and ended in the same round, q^{\sim} will be updated three times, and we acquire $q^{\sim} = 12.73$ in round 2.

Table 3. 7: Component3 Apply RR (quantum calculations round2)

Task	$B_{ij} = B_i(j-1) - q_i(j-1)$	$(B_{ij} + q_i(j-1))^2$	$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(B_{ij} + (-1)^{1-\alpha} \cdot q_{i(j-1)})^2}$
Task6	15-13.56=1.44	(1.44-13.56) ² =146.8	q62 =9.79
Task3	23-13.52= 9.48	(9.48-13.52) ² = 16.32	q32 =10.32
Task5	30-13.51= 16.49	(16.49-13.51) ² = 8.88	q52=10.82

Time-based on the table (3.6)			
Task6	Task3	Task5	
70.59	80.38	90.7	101.52

Table 3. 8: Component 4 the Responding, Waiting and Turnaround Times by HARSQ Compared to Traditional SJF and RR

Tasks	responding time (ms)			Waiting time (ms)			Turnaround time (ms)		
	HARSQ	SJF	RR	HARSQ	SJF	RR	HARSQ	SJF	RR
Task1	31.56	18	0	31.56	18	48	43.56	30	60
Task2	0	0	5	0	0	30	8	8	38
Task3	43.56	45	10	67.7	44	65	90.7	67	88
Task4	8	8	15	8	6	38	18	16	48
Task5	57.8	68	20	78.28	65	68	108.28	95	98
Task6	18	30	25	65.30	26	60	80	41	75
Average time	26.486	28.166	12.5	41.806	26.5	52.5	58.09	42.83	67.833

Table (3.8) validates that the HARSQ responding time is less compared to traditional SJF and RR algorithms, but with the higher turnaround and waiting time. We can conclude by demonstrating that HARSQ is the complementary balance point among SJF and RR, in which we attempted to reduce RR and SJF starvation problems.

3.4. Settings Simulation Environment

A suggested hybrid approach applied and verified in two dissimilar settings C# and CloudSim simulation setting.

3.4.1. Visual Studio Simulation Environment

As an integrated development tool, the C# was guided by C# 2013 Express Editions. WPF (Windows Foundation Presentation) and XML using into the environment which is a user interface. Microsoft developed the WPF framework based on the NET framework 3.0 architecture basis.

Allowing the programmer to manipulate complex interfaces and controls quickly is the main feature of WPF (Jeyarani, Ram, and Nagaveni 2010). Windows 7 Home Edition x64 with RAM (4 GB), 8 MB cache, and Intel Core i5 2nd generation were used for simulation.

3.4.2. CloudSim Simulation Environment

A tool to simulate and to model cloud computing environments called CloudSim. SimJava provides an advanced package library which can be implemented in a Windows operating (Garg *et al.*, 2011). There are many original features of CloudSim we illustrated it as the following points:

1. Provide virtual machine simulation of cloud computing environments
2. Offer platform allocation strategies on resources.
3. Simulate affordable network connections.

Moreover, specific features of CloudSim include helping to create and control different data centres by providing a virtual engine and sharing virtualization services. Besides, it resists the change between the space share and the timeshare. Quickly improve algorithms and methods of cloud computing with the help of CloudSim environmental tools.

User code, GridSim, SimJava, and CloudSim are four layers of CloudSim simulator from lowest to up (Garg *et al.*, 2011). SimJava is the leading simulation engine. Carrying out the central roles of the top-level simulation model is a SimJava task, such as query as well as handling events and body system elements (virtual machines, data centres, clients, services, and agents).

Using GridSim, the Cloudsim level performance expands the basic functionality provided. The outfit layer provides virtual data centre control interfaces that count RAM and the virtual machine.

The simulated cloud computing setting involves a user, a single data centre, and a broker to assess the suggested model. CloudSim provides a cloud-based interface to be created and to perform a series of experiments.

For the virtual space policy of virtual machines, it is used to allocate the cloudlets (tasks) so that the tasks are executed sequentially in each Virtual Machine (VM). In contrast, for hosts, the virtual FCFS algorithm uses the VM specialty. The number of inward tasks or the size of the queue did not disturb the individual task units and the execution time using this policy because each unit has its core task and the suggested approach is a non-rudimentary technique.

In the CloudSim setting, assessment trials were accomplished in 3 suitcases using one VM, two VMs and 3 VMs. These three datasets were similarly reprocessed for testing.

Evaluation experiments in the CloudSim environment were conducted in three cases using (one, two, and three) VM. For selection, the three datasets were reused.

Figure (3.2) summarizes the simulation parameters and settings used in CloudSim experiments.

```
private static List<Cloudlet> createCloudlet(int userId, int cloudlets){
    // Creates a container to store Cloudlets
    LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

    //cloudlet parameters
    long length = 1000;
    long fileSize = 300;
    long outputSize = 300;
    int pesNumber = 1;
    UtilizationModel utilizationModel = new UtilizationModelFull();

    Cloudlet[] cloudlet = new Cloudlet[cloudlets];

    for(int i=0;i<cloudlets;i++){
        Random r= new Random();
        cloudlet[i] = new Cloudlet(i, length+r.nextInt(2000), pesNumber, fileSize, outputSize, utilizationModel);
        // setting the owner of these Cloudlets
        cloudlet[i].setUserId(userId);
        list.add(cloudlet[i]);
    }

    return list;
}
```

Figure 3. 2 Cloudsim Simulation Parameters

3.5. Metrics Performance

The subsequent metrics are measured through the assessment progression

1. Wait time: Average time a process spent in the run queue.
2. Response Time: Averaged period elapsed from the time of process submission until useful output obtained.
3. Turnaround Time: Averaged period elapsed from when a progression was submitted when it was accomplished.
4. Throughput: Number of sequences performed / time unit

CHAPTER FOUR :EXPERIMENTS AND DISCUSSIONS

4.1. Overview

In this chapter, experimental result and evaluation of the hybrid approach using RR and SJF by quantum, dynamics are presented. First, produce an examination for different scheduling algorithms to explore which algorithm is more appropriate, helping for building our hybrid approach. Second, evaluation and dissection of the model designed by the researcher. We described all experimental steps for each component using the c# framework and cloudsim environment for task scheduling. Finally, the result was compared using a different algorithm.

4.2. Prepare experimental environment

For assessment Hybrid approach, we used three different datasets that have been used within trying the six different methods [RR (Quantum=8), SJF, RRS and FCFS, MAX-MIN, MIN-MIN, RRS and SRTF].

The first step is to prepare the C# experimental environment, as shown in figure (4.1).

```
1
2 Microsoft Visual Studio Solution File, Format Version 11.00
3 # Visual Studio 2010
4 Project("{FAE04EC0-301F-11D3-BF48-00C04F79EFBC}") = "WindowsFormsApplication2", "WindowsFormsApplication2\WindowsFormsApplication2.csproj",
5 EndProject
6 Global
7     GlobalSection(SolutionConfigurationPlatforms) = preSolution
8         Debug|x86 = Debug|x86
9         Release|x86 = Release|x86
10    EndGlobalSection
11    GlobalSection(ProjectConfigurationPlatforms) = postSolution
12        {5B6B62BD-6113-4F99-B653-256707E4F9A2}.Debug|x86.ActiveCfg = Debug|x86
13        {5B6B62BD-6113-4F99-B653-256707E4F9A2}.Debug|x86.Build.0 = Debug|x86
14        {5B6B62BD-6113-4F99-B653-256707E4F9A2}.Release|x86.ActiveCfg = Release|x86
15        {5B6B62BD-6113-4F99-B653-256707E4F9A2}.Release|x86.Build.0 = Release|x86
16    EndGlobalSection
17    GlobalSection(SolutionProperties) = preSolution
18        HideSolutionNode = FALSE
19    EndGlobalSection
20 EndGlobal
```

Figure 4. 1Prepare experimental environment with C#

4.3. Comparing scheduling algorithms

This section is existing a conclusion of reviews that used scheduling approaches for assessment, by using three dissimilar datasets that have been used within testing the six evaluated strategies [RR (Quantum=8), SJF, RRS, and FCFS, MAX-MIN and MIN-MIN, RRS and SRTF].

Every dataset involves one hundred percent produced haphazardly and dynamically mixed to produce unequal randomization looking for a perfect benchmark to the assessment resolution. Five tasks were represented as Task1, Task2, Task3, Task4, Task5, and every task was described via its arrival time and burst time, as shown in Table (4.1).

Table 4. 1: Dataset used on experiment (Component 1)

Task	Dataset1		Dataset2		Dataset3	
	Burst Time	Arrival Time	Burst Time	Arrival Time	Burst Time	Arrival Time
T1	50	0	250	0	30	0
T2	100	1	200	1	300	1
T3	150	2	150	2	200	2
T4	200	3	100	3	20	3
T5	250	4	50	4	10	4

Figure 4. 2 Result of Time using six scheduling algorithms.

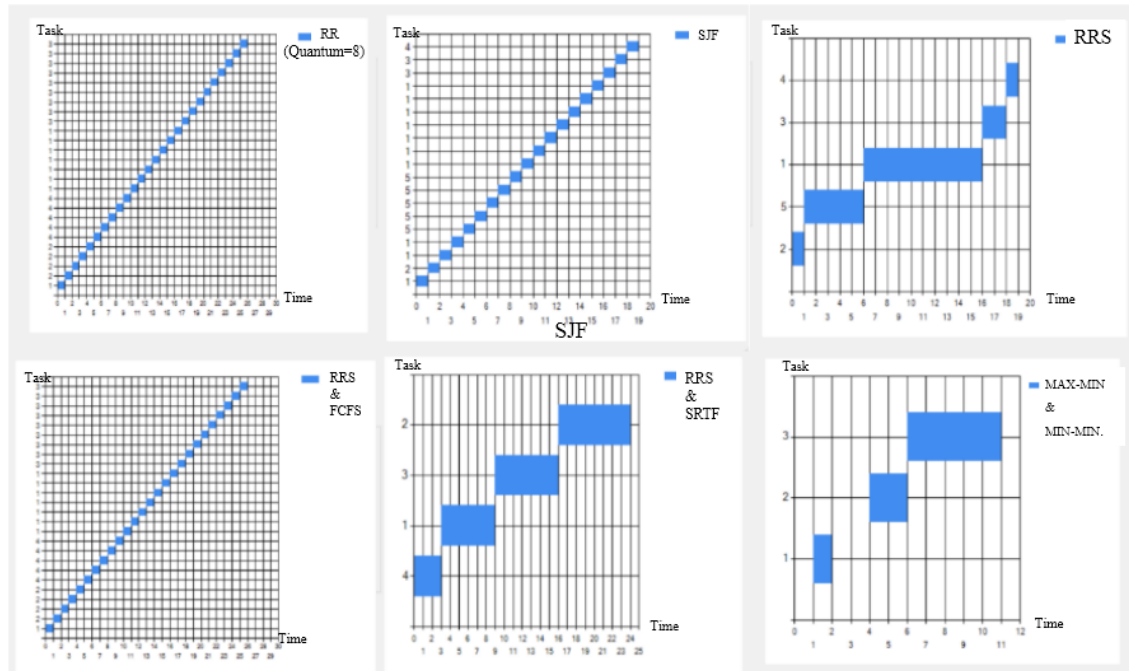


Table 4. 2 the evaluation results of the six implemented algorithms

Dataset	Dataset1			Dataset2			Dataset3		
	Turnar ound Time	Waiting Time	Response Time	Turnar ound Time	Waiting Time	Response Time	Turna round Time	Waiting Time	Response Time
RRS and FCFS	510	380	99	490	380	165	210	110	47
RRS and SRTF	510	380	112	500	360	150	200	90	40
SJF	300	190	195	430	290	290	190	85	65
RR (Quantum=8)	520	395	22	520	400	15	220	150	26
MAX-MIN	310	190	35	410	295	50	170	50	43
MIN-MIN	290	170	29	400	270	40	185	65	37

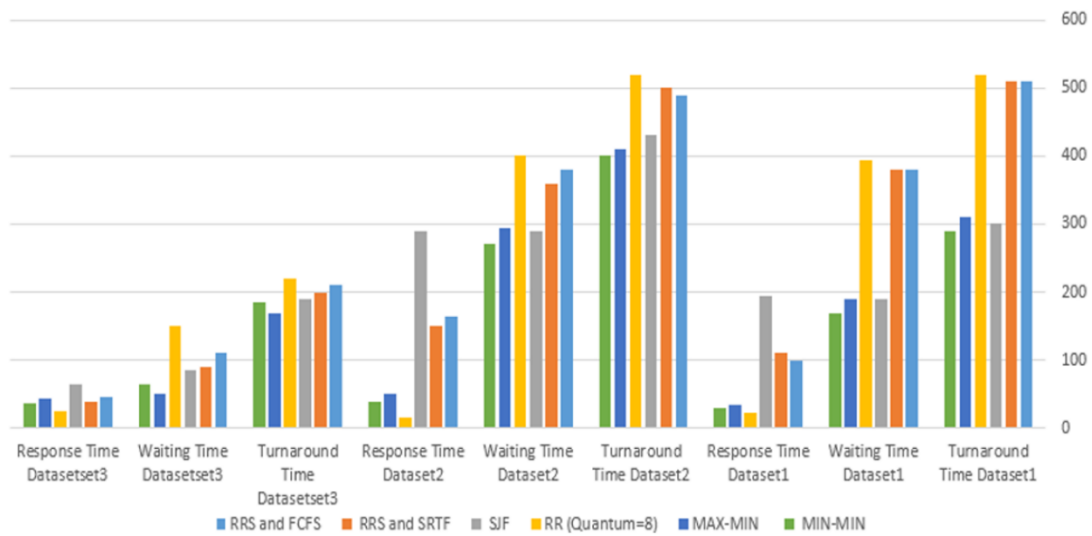


Figure 4. 3 represents the evaluation results of the six implemented algorithms

By comparing the result of algorithms, we found that RR, RRS, and SRTF counted the maximum turnaround period. Moreover, SJF, Min-Min, and Max-Min attained the lowest waiting time, whereas standard RR with quantum equal eight reached the most impoverished waiting time. Confirmed, similarly in this figure (4.2) that the Min-Min procedure had been evidenced highness overall in datasets one and two while the Max-Min attested its majesty in dataset three. Finally, one could determine from the evaluation of experimental that the Min-Min effectively accomplished and earned the competitive in contrast to the six applied standard approaches and hybrids in lowered the times of the turnaround and waiting.

The time of a response in the cloud setting symbolizes a rapid response to users of services in which it was reflected one of the most critical metrics in this heterogeneous setting.

Also, outcomes of SJF have the top response time. However, RR succeeded in the tiniest response time. Max-Min and Min-Min have an adequate response time, RRS&FCFS and RRS&SRTF have an adjacent rank authorizing the benefit of the Min-Min and Max-Min approach comprehensive datasets. Table (4.2) and Figure (4.3) symbolizes the results of the evaluation of the 6 applied approaches.

We found the RR algorithm more appropriate to construct the new version of this hybrid model regarding the problem of starvation. However, the result shows the simulation in previously graphs max-min, min-min have a perfect turnaround and waiting period. Still, they affected from a starvation problem, FCFS, SRTF do not display a perfect hybrid with

RR due to they have been a high waiting and turnaround time. RR standard approach recognized as one of the chief advantages further its equality is the good Response time. For the service of quality requests, we weighted the throughput for six competitive approaches. We defined the throughput as the number of completed tasks divided by a deep average of the whole time. Max-Min and Min-Min attained the equal supreme performance in datasets one while standard RR by quantum equal eight and SJF attained the poorest throughput. Confirmed, likewise in this figure that the Min-Min approach had been evidenced highness inclusive second and third dataset directed by Max-Min. As a final point, one could conclude from the trail evaluation that the Min-Min accomplished efficiently and obtained the competition besides the six applied standard approaches and the two-hybrid approaches in decreasing the turnaround, waiting time, and throughput, even though RR saved its highness concerned through response time.

4.4. A hybrid approach using RR and SJF by Quantum Dynamic

4.4.1. Component three: Modified RR

The modified RR approach has been dependent on the dynamics of the time quantum tactic wherever the system adjusts time. Quantum was conferring from the burst time of processes established in the prepared queue. We compared modified RR with static RR and additional than literature approach, Which already has been whole of them and prepared specific modifications on standard RR to defeat the fixed the problem of the time quantum value in RR. We test improved RR in the C#.

We applied benchmark dataset for two test to explore more about the parameter that affect the performance of modified RR; first test is for define the best time quantum with task arrived randomly, by second test we reorder arrive time with increase and decrease order. it is clear that Modified RR shows good performance concerning to waiting and turnaround time in first test, and second test.

4.4.2. First test

Three different datasets have been used over testing the four evaluated approaches [RR (Quantum=20), NA and Modified RR, SARR]. Benchmark income from (Noon, Kalakech, and Kadry 2011) for the evaluation. Dataset showed on a table (4.3).

Table 4. 3 first test dataset

Task	Case 1		Case 2		Case 3	
	Burst time	Zero Arrival time	Zero Burst time	Arrival time	Burst time	Arrival time
Task1	20ms			18	18	10
Task2	40ms			70	70	14
Task3	60ms			74	74	70
Task4	8ms			80	80	120

After Experiment, it is pure that improved RR illustrations perfect performance regarding waiting and turnaround times in the first 3 cases.

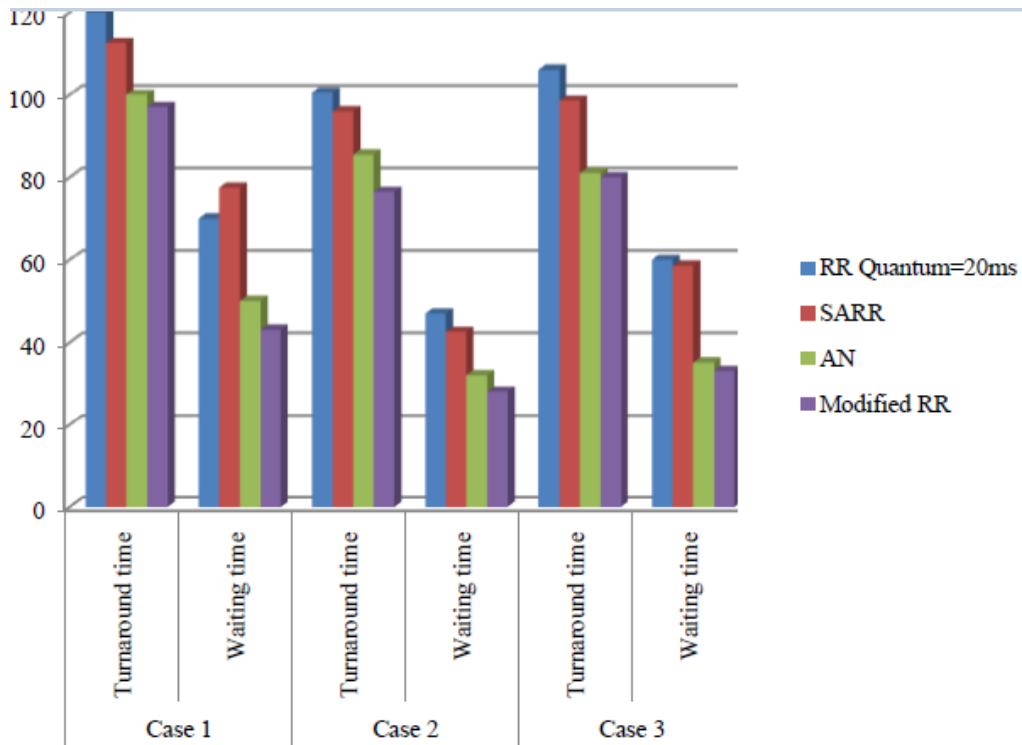


Figure 4. 4 comparison result for first test

4.4.3. Second test

This testing contains several input and output factors. The input factors contain burst, arrival time, and the number of tasks. The output factors contain the average waiting time, average turnaround. Results Took from the proposed improved algorithm effectively could work with a huge number of data. The test was separated into 2 cases; a head case arrival time

when it is 0 since the specified tasks, and another case arrival time was different for tasks. Five procedures were taken into consideration Task1, Task2, Task3, Task4 and Task5 was occupied as a benchmark from (Nayak, Kumar Malla, and Debadarshini 2012) for comparison peruse.

Table 4. 4 second test dataset

Task	Case 1: With Zero Arrival Time			Case 2: Without Zero Arrival Time					
	Increasing Order	Decreasing Order	Random Order	Increasing Order		Decreasing Order		Random Order	
	Burst Time	Burst Time	Burst Time	Arrival Time	Burst Time	Arrival Time	Burst Time	Arrival Time	Burst Time
T1	30	77	80	0	14	0	80	0	65
T2	34	54	45	2	34	2	74	1	72
T3	62	45	62	6	45	3	70	4	50
T4	74	19	34	8	62	4	18	6	43
T5	88	14	78	14	77	5	14	7	80

Figure (4.5) shows modified RR has a suitable turnaround and waiting time with no time of arrival. Except arrival time despite the dispersal of the performance rank of modified RR, nevertheless, it is quite a suitable performance when compared with IRR and RR

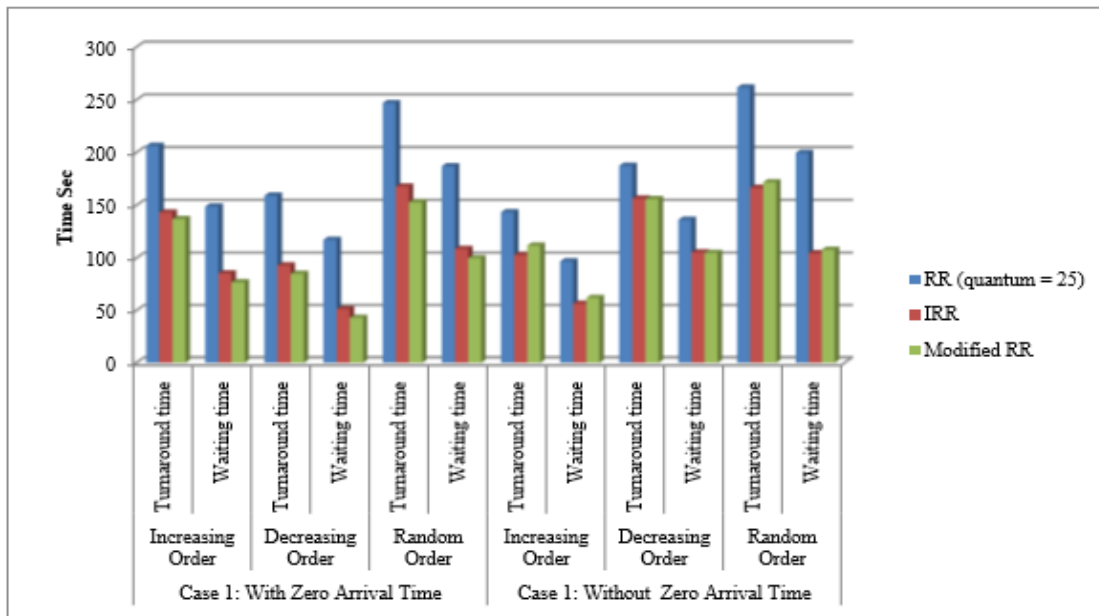


Figure 4. 5 second comparison results.

The latest test was made within the CloudSim setting by a randomized dataset of ten cloudlets by random long burst and arrival time produced by the setting shown in Table (4.5) to discover the impact of the suggested algorithm to reduce the problem of starvation

Table 4.5 ten cloudlets by random long burst and arrival time produced by the environment

Cloudlet ID	VM ID	Arrival	Burst time
0	0	0	12
5	0	8	13
2	0	5	44
7	0	11	92
4	0	8	101
8	0	13	144
3	0	7	157
9	0	15	158
6	0	19	179
1	0	5	210

4.4.4 Last test

It was noted the cloudlets one, three, six, eight and nine burst time is extended that determine these cloudlets will occur by starvation if the SJF schedulers were practical and also will affect if the RR quantum was small. Through the suggested algorithms with its two versions, we tried to balance between reducing cloudlets waiting time and rising quantum value. Furthermore, we attempted to attain equality in choosing cloudlets for execution while having two short cloudlets from Q1 and one lengthy cloudlet from Q2.

4.5. A proposed Hybrid approach in C#

The suggested approach was likened with the typical RR and SJF. Using the exact three dissimilar datasets in Table 1. Chief performance benchmarks were used to link approaches were: waiting time, response time, turnaround time and throughput. The unit of time was characterized by the throughput metric was designated haphazardly.

To the extent the ideal case in the planned model, the researcher conducted three tests by shifting a number of choosing tasks alternatively between two sub-queues (Q1 and Q2):

First Situation: Selected one tiny job from the first queue (Q1) then and there selected one huge task from the second queue (Q2), (1 small (Q1): 1 second (Q2)). Second situation: Chose two small tasks from a small queue(Q1) then chose one large task from a large queue (Q2), (2 small(Q1): 1-second queue (Q2)). Third situation: Selected one tiny task from the first queue (Q1) then selected two huge tasks from a second queue (Q2), (1 small (Q1): 2 large (Q2)). Improved RR was also produced by a monitoring period of quantum via a formula that explains it calculated dynamically in addition to linked by preceding 3 cases of recommended practice.

4.6. The Quality of Service Measurement

As one of the services of the quality supplies, we weighted throughput of suggested methods and compared it with the conventional RR and SJF.

Throughput weighted by the number of the whole task which has finished by an average of complete time.

Figure (4.6) shows the different throughputs attained to, different practical tests.

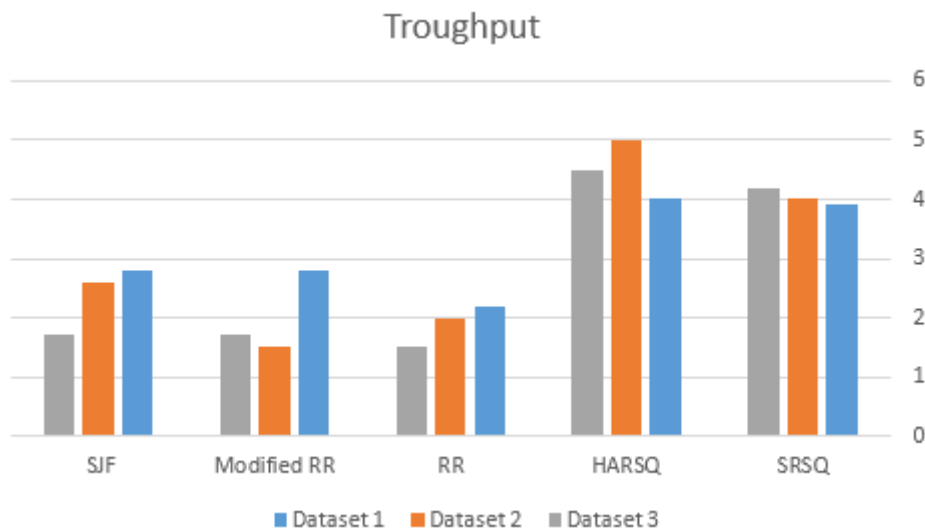


Figure 4. 6 different throughputs achieved

From the test assessment. It could be determined that the HARSQ achieves effectively via a predefined fixed assessment as the quantum of time in the turnaround and waiting time somewhat SJF, RR and using the suggested an adaptive equation, which in transformation completed improved in response time.

Lastly, Figure (4.6) denotes the assessment outcomes of matching HARSQ and SRSQ versus a hybrid of Round Robin Scheduling & First Come First Serve (RRS&FCFS), Round Robin Scheduling and & Shortest Remaining Time First (RRS&SRTF) in that the outcomes displayed the HARSQ is added capable than the other in turnaround and waiting time averages overall used data sets.

Table 4. 6 First Experimentation Results using 3 different VM

Technique		Single virtual machine			Two virtual machines			Three virtual machines		
		Response Time	Waiting Time	Turnaround Time	Response Time	Waiting Time	Turnaround Time	Response Time	Waiting Time	Turnaround Time
Modified RR		126.6	126.6	252.8	7.2	7.2	153	7.2	7.2	120.6
HARSQ (quantum =8)	1 small(Q1): 1 large (Q2)	7.6	98	210	4.6	13.4	125.4	5.4	11	123
	2 small (Q1): 1 large (Q2)	7.2	87.2	199.2	4.2	33.8	145.8	5.8	30	142
	1 small(Q1): 2 large (Q2)	20.2	133.2	245.4	19	53.4	165.4	182.6	211.4	323.4
HARSQ	1 small(Q1): 1 large (Q2)	92.4	98.2	212	7.2	7.2	152.8	7.2	7.2	120.6
	2 small (Q1): 1 large (Q2)	112.4	112.4	234.6	7.2	7.2	152.8	7.2	7.2	120.6
	1 small(Q1): 2 large (Q2)	92.4	98.2	212	7.2	7.2	152.8	7.2	7.2	120.6

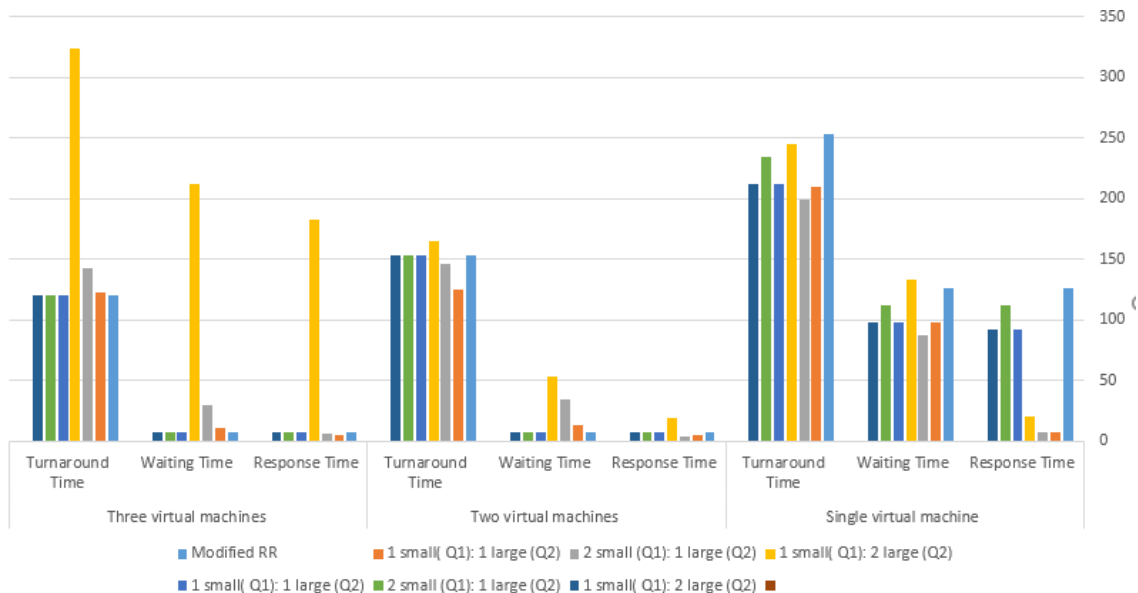


Figure 4. 7 Result of testing HARSQ on 3 different VM

4.7 Discussion constructed on first dataset illustrations: -

response time: scheduled solo virtual machines were the greatest in HARSQ (quantum = 8) whereas it was steady in HARSQ an improved RR, on two virtual machines HARSQ (quantum = 8) in the second case had the peak bar, on three virtual machine HARSQ in the first case and second case obtainable shorter response time, in this research response time is acceptable in the planned model as it was in improved RR.

Waiting time: scheduled single virtual machines were great in HARSQ but, HARSQ (quantum = 8) ; nonetheless, HARSQ and improved RR provided approximate waiting time. Scheduled twofold virtual machines HARSQ and improved RR have better levels from HARSQ (quantum = 8). On three virtual machines, we obtained approximately similar rates that were attained from dual virtual machines.

Turnaround time: scheduled a single virtual machine, HARSQ and improved RR formed adequate turnaround time related to HARSQ (quantum = 8). These ranks of turnaround time remain over 2 and 3 virtual machines.

Table **below**, represent experiments applying on other dataset using one, two, three virtual machines. in HARSQ (quantum = 8) set time quantum to be 8 while it change dynamically in HARSQ and Modified RR.

Table 4. 7 Second Experimentation Results using 3 different VM

Technique		Single virtual machine			Two virtual machines			Three virtual machines		
		Response Time	Waiting Time	Turnaro und Time	Response Time	Waiting Time	Turnaro und Time	Response Time	Waiting Time	Turnaro und Time
Modified RR		145.4	180.6	330.6	31.4	31.4	191.2	31.6	31.6	191.4
HARSQ (quantum =8)	1 small(Q1): 1 large (Q2)	8.8	200	350	6.4	98.8	248.4	16.2	71.4	221.4
	2 small (Q1): 1 large (Q2)	10.4	129.4	280.4	27.4	66.4	216.4	6.8	45.8	195.8
	1 small(Q1): 2 large (Q2)	13.4	193.6	343.6	26.8	113.2	263.2	6.8	56	206
HARSQ	1 small(Q1): 1 large (Q2)	150.8	170.6	320.8	31.2	31.6	191	31.2	31.2	190.6
	2 small (Q1): 1 large (Q2)	150.2	200.2	360.02	31.6	31.6	191	31.2	31.2	190.6
	1 small(Q1): 2 large (Q2)	150.8	170.6	320.8	31.2	31.6	191	31.2	31.2	190.6

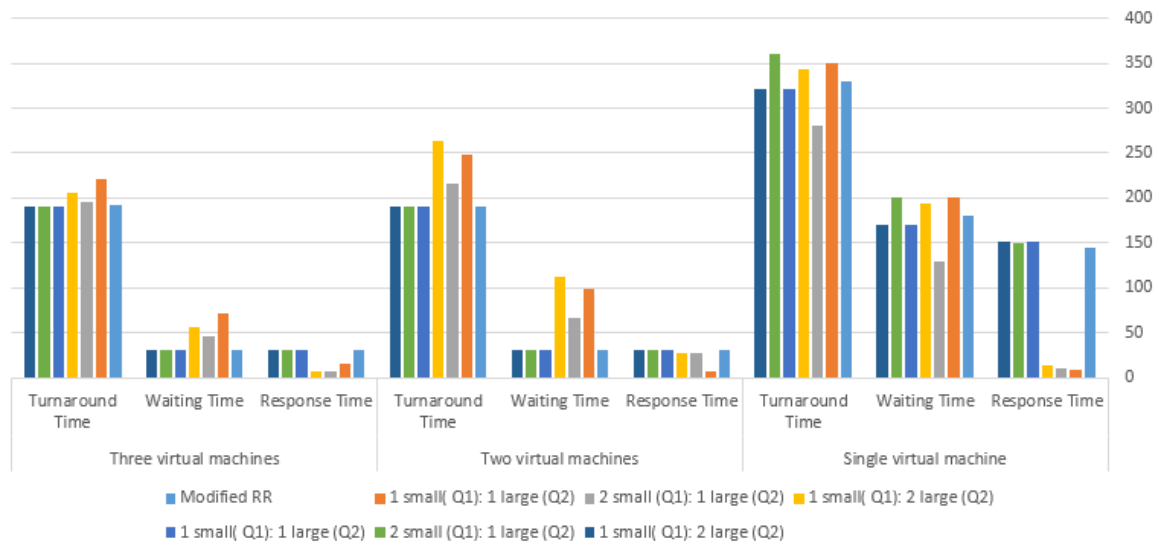


Figure 4.8 Second Experimentation Results using 3 different VM

Response time: HARSQ (quantum 8) on an single virtual machine is the greatest even though HARSQ and improved RR are closets, everyone to new in rising values; on two virtual machines all bars had been approximate stages unless HARSQ in stage one which has been the lowermost bar; in 3 virtual machines HARSQ offered earlier response time.

Waiting time: on only virtual machine HARSQ (quantum 8) in stage two had been a suitable stage, but HARSQ and improved RR offered approximate waiting time. In 2 virtual machines, HARSQ and improved RR had been enhanced levels from HARSQ (quantum 8), and this difference remains above three virtual machines.

Turnaround time: in an only virtual machine, HARSQ (quantum 8) in stage two made effective turnaround time compared to HARSQ and improved RR. In 2 virtual machines, HARSQ (quantum 8) in stage three was the greatest one, whereas HARSQ and improved RR seem in equivalent stages in 3 virtual machines.

Decreasing the waiting time point the average time a cloudlet uses in the run queue is decreased that have been reduced cloudlet starvation. HARSQ attained the waiting time was reduced. Thus starvation within that has been the two sub-queues Q1 and Q2 which Q1 for petite tasks and Q2 for the recreation based on the median of a task. Numerous assessments and testing have been done by the assistants to discovery the excellent approaches to choosing jobs obtain from Q1 and Q2 to be allocated to resources and lastly establish that as simplified in the approach that has been two jobs from Q1 and 1 task from Q2, really that has a perfect influence in decreasing job starvation.

A part of the simulation outcomes, it is noticeable that HARSQ and HARSQ (quantum 8) have attained a respectable performance likened to the basic RR and SJF in addition to hybrid RRS&SRTM and RRS&FCFS in response and waiting time. It is similarly apparent that HARSQ dropping waiting and response times, whereas HARSQ (quantum 8) surpasses in decreasing turnaround time. We are able to assure that the suggested approach in its dual versions (HARSQ and HARSQ (quantum 8)) has attained a good saving in the waiting time of each one task and in the inclusive waiting average, from which we can say that it leads to reducing task starvation, which is one of our first urgencies.

The tests results had been shown that dynamicity in task quantum has a good influence on dropping task waiting and response time. Even though the dynamicity in separated task quantum from round to round had, an acceptable influence on diminishing turnaround time. As a result, we be able to see the HARSQ working as the equally point with influence, waiting and starvation decline exclusively in tasks have long burst times and with equal performance in turnaround to SJF, RR, RSS&SRTM, RRS&FCFS and TSPBRR. impossible.

CHAPTER FIVE: CONCLUSIONS and FUTURE WORKS

5.1. Conclusions

Task scheduling is one of the chief research areas in cloud computing wherever dissimilar scheduling algorithms are presented fluctuating from conventional SJF, MQ and RR methods to numerous hybrids as Min-Min, RRS&FCFS and Min-Max. Numerous tests of enhancing a scheduler as well as SJF with a few waiting and therefore, fewer task starvation had been made. Many researchers had advanced a hybrid of SJF & RR to acquire advantage of good

In this work, a hybrid of SJF&RR algorithm was suggested, unless with two sub-queues: the first for closely a little job and the others were long. The tasks were allocated approach equilibrium between the two sub-queues to reducing task starvation, particularly in tasks with closely long burst time. The suggested algorithm was constructed on a dynamic task quantum counting procedure depend on the median of the presented tasks and the task quantum in the preceding round of the RR scheduler.

The suggested approaches were simulated in 2 dissimilar forms of SJF&RR within dynamic quantum (HARSQ) and SJF&RR through static quantum (HARSQ) in 2 diverse circumstances a C# program and a CloudSim setting with one, two and three virtual machines. Mutually versions of suggested approach displayed a good performance in minimizing response and waiting period and an equal performance within turnaround time to state of the art.

5.2. The Work of the Future

In Future, we hope to proceed tests in the discovery the improved task quantum approach that equilibrium among the dynamic quantum values and static to attain well decline in waiting as a result in decrease task starvation.

References

- Alworafi, Mokhtar A., Atyaf Dhari, Sheren A. El-Booz, Aida A. Nasr, Adela Arpitha, and Suresha Mallappa. 2019. "An Enhanced Task Scheduling in Cloud Computing Based on Hybrid Approach." *Lecture Notes in Networks and Systems* 43: 11–25.
https://doi.org/10.1007/978-981-13-2514-4_2.
- Attaran, Mohsen, and Jeremy Woods. 2019. "Cloud Computing Technology: Improving Small Business Performance Using the Internet." *Journal of Small Business and Entrepreneurship* 31 (6): 495–519. <https://doi.org/10.1080/08276331.2018.1466850>.
- Boniface, Michael, Bassem Nasser, Juri Papay, Stephen C. Phillips, Arturo Servin, Xiaoyu Yang, and Zlatko Zlatev. 2010. "Platform-as-a-Service Architecture for Real-Time Quality." *Of Service Management in Clouds* & Quot;, *Fifth International Conference on Internet and Web Applications and Services, Iciw & Apos;10*.
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.301.1856>.
- Brodkin, Jon. 2008. "Gartner: Seven Cloud-Computing Security Risks." *Infoworld* 2008: 1–3.
- Černý, Pavol, Edmund M. Clarke, Thomas A. Henzinger, Arjun Radhakrishna, Leonid Ryzhyk, Roopsha Samanta, and Thorsten Tarrach. 2017. "From Non-Preemptive to Preemptive Scheduling Using Synchronization Synthesis." *Formal Methods in System Design* 50 (2–3): 97–139. <https://doi.org/10.1007/s10703-016-0256-5>.
- Cusumano, Michael. 2010. "Cloud Computing and SaaS as New Computing Platforms." *Communications of the ACM* 53 (4): 27–29. <https://doi.org/10.1145/1721654.1721667>.
- Dave, Yash P., Avani S. Shelat, Dhara S. Patel, and Rutvij H. Jhaveri. 2015. "Various Job Scheduling Algorithms in Cloud Computing: A Survey." *2014 International Conference on Information Communication and Embedded Systems, ICICES 2014*, no. 978.
<https://doi.org/10.1109/ICICES.2014.7033909>.
- Demchenko, Yuri, and Cees De Laat. 2011. "Defining Generic Architecture for Cloud Infrastructure as a Service Model." *Proceedings of Science*, 1–10.

<https://doi.org/10.22323/1.133.0115>.

- Frijns, R. M. W., S. Adyanthaya, S. Stuijk, J. P. M. Voeten, M. C. W. Geilen, R. R. H. Schiffelers, and H. Corporaal. 2014. "Timing Analysis of First-Come First-Served Scheduled Interval-Timed Directed Acyclic Graphs," 1–6.
<https://doi.org/10.7873/date.2014.301>.
- Ganapathi, Archana, Yanpei Chen, Armando Fox, Randy Katz, and David Patterson. 2010. "Statistics-Driven Workload Modeling for the Cloud." *Proceedings - International Conference on Data Engineering*, 87–92. <https://doi.org/10.1109/ICDEW.2010.5452742>.
- Garg, Saurabh Kumar, and Rajkumar Buyya. 2011. "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations." *Proceedings - 2011 4th IEEE International Conference on Utility and Cloud Computing, UCC 2011*, no. Vm: 105–13.
<https://doi.org/10.1109/UCC.2011.24>.
- Hausmans, Joost P.H.M., Stefan J. Geuns, Maarten H. Wiggers, and Marco J.G. Bekooij. 2013. "Dataflow Analysis for Multiprocessor Systems with Non-Starvation-Free Schedulers." *Proceedings of the 16th International Workshop on Software and Compilers for Embedded Systems, M-SCOPES 2013*, 13–22.
<https://doi.org/10.1145/2463596.2463603>.
- Hwang, K, G Fox, and J Dongarra. 2012. "Distributed and Cloud Computing Parallel Computing and Programming Enviroments," 1–57.
<https://doi.org/10.1017/CBO9781107415324.004>.
- Jelassi, Mariem, Cherif Ghazel, and Leila Azzouz Saidane. 2017. "A Survey on Quality of Service in Cloud Computing." *2017 3rd International Conference on Frontiers of Signal Processing, ICFSP 2017*, no. October: 63–67.
<https://doi.org/10.1109/ICFSP.2017.8097142>.
- Jeyarani, R., R. Vasanth Ram, and N. Nagaveni. 2010. "Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment." *CCGrid 2010 - 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing*, 585–86.
<https://doi.org/10.1109/CCGRID.2010.94>.

- Jula, Amin, Elankovan Sundararajan, and Zalinda Othman. 2014. "Cloud Computing Service Composition: A Systematic Literature Review." *Expert Systems with Applications* 41 (8): 3809–24. <https://doi.org/10.1016/j.eswa.2013.12.017>.
- Kaur, Arshjot, and Supriya Kinger. 2014. "A Survey of Resource Scheduling Algorithms in Green Computing." *International Journal of Computer Science and Information Technologies* 5 (4): 4886–90.
- Kaur, Simranjit, Pallavi Bagga, Rahul Hans, and Harjot Kaur. 2019. "Quality of Service (QoS) Aware Workflow Scheduling (WFS) in Cloud Computing: A Systematic Review." *Arabian Journal for Science and Engineering* 44 (4): 2867–97. <https://doi.org/10.1007/s13369-018-3614-3>.
- Kurdi, Heba, Ebtesam Aloboud, Sarah Alhassan, and Ebtehal T. Alotaibi. 2014. "An Algorithm for Handling Starvation and Resource Rejection in Public Clouds." *Procedia Computer Science* 34: 242–48. <https://doi.org/10.1016/j.procs.2014.07.018>.
- Lakhani, Jignesh, and Hitesh A Bheda. 2013. "An Approach to Optimized Resource Scheduling Using Task Grouping in Cloud." *International Journal of Advanced Research in Computer Science and Software Engineering* 3 (9): 2277–128.
- Lakhani, Jignesh, Hitesh A Bheda, Arslan Munir, Prasanna Kansakar, Samee U. Khan, Katarina Stanoevska-Slabeva, Thomas Wozniak, et al. 2020. "A Hybrid Approach for Scheduling Applications in Cloud Computing Environment." *International Journal of Advanced Research in Computer Science and Software Engineering* 159 (9): 33–42. <https://doi.org/10.1007/978-3-642-05193-7>.
- Li, Jiayin, Meikang Qiu, Jian Wei Niu, Yu Chen, and Zhong Ming. 2010. "Adaptive Resource Allocation for Preemptable Jobs in Cloud Systems." *Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, ISDA'10*, 31–36. <https://doi.org/10.1109/ISDA.2010.5687294>.
- Lu, Chih Wei, Chih Ming Hsieh, Chih Hung Chang, and Chao Tung Yang. 2013. "An Improvement to Data Service in Cloud Computing with Content Sensitive Transaction Analysis and Adaptation." *Proceedings - International Computer Software and*

- Applications Conference*, 463–68. <https://doi.org/10.1109/COMPSACW.2013.72>.
- Mezgár, István, and Ursula Rauschecker. 2014. “The Challenge of Networked Enterprises for Cloud Computing Interoperability.” *Computers in Industry* 65 (4): 657–74. <https://doi.org/10.1016/j.compind.2014.01.017>.
- Nayak, Debashree, Sanjeev Kumar Malla, and Debashree Debadarshini. 2012. “Improved Round Robin Scheduling Using Dynamic Time Quantum.” *International Journal of Computer Applications* 38 (5): 34–38. <https://doi.org/10.5120/4607-6816>.
- Noon, Abbas, Ali Kalakech, and Seifedine Kadry. 2011. “A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average,” no. June. <http://arxiv.org/abs/1111.5348>.
- Patel, Ahmed, Mona Taghavi, Kaveh Bakhtiyari, and Joaquim Celestino. 2012. “Taxonomy and Proposed Architecture of Intrusion Detection and Prevention Systems for Cloud Computing.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7672 LNCS: 441–58. https://doi.org/10.1007/978-3-642-35362-8_33.
- Patel, Swachil, and Upendra Bhoi. 2013. “Priority Based Job Scheduling Techniques In Cloud Computing: A Systematic Review.” *International Journal of Scientific & Technology Research* 2 (11): 147–52.
- Raj, Gaurav, Dheerendra Singh, and Abhay Bansal. 2013. “Load Balancing for Resource Provisioning Using Batch Mode Heuristic Priority in Round Robin (PBRR) Scheduling.” *IET Conference Publications* 2013 (647 CP): 308–14. <https://doi.org/10.1049/cp.2013.2333>.
- Rajput, Ishwari Singh, and Deepa Gupta. 2013. “A Priority Based Round Robin CPU Scheduling Algorithm for Real Time Systems.” *Journal of Advanced Engineering Technologies* 2 (3): 120–124.
- Santra, Soumen, Hemanta Dey, Sarasij Majumdar, and Gauri Shankar Jha. 2014. “New Simulation Toolkit for Comparison of Scheduling Algorithm on Cloud Computing.”

2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014, 466–69.

<https://doi.org/10.1109/ICCICCT.2014.6993007>.

Singh, Sukhpal, and Inderveer Chana. 2016. “A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges.” *Journal of Grid Computing* 14 (2): 217–64.
<https://doi.org/10.1007/s10723-015-9359-2>.

Tiwari, Devanshu. 2014. “An Efficient Hybrid SJF and Priority Based Scheduling of Jobs in Cloud Computing” 2 (4).

Venters, Will, and Edgar A. Whitley. 2012. “A Critical Review of Cloud Computing: Researching Desires and Realities.” *Journal of Information Technology* 27 (3): 179–97.
<https://doi.org/10.1057/jit.2012.17>.

Villamizar, Mario, Oscar Garces, Lina Ochoa, Harold Castro, Lorena Salamanca, Mauricio Verano, Rubby Casallas, et al. 2016. “Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures.” *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, 179–82.
<https://doi.org/10.1109/CCGrid.2016.37>.

Wu, Xian, and Peide Qian. 2013. “Towards the Scheduling of Access Requests in Cloud Storage.” *Proceedings of the 8th International Conference on Computer Science and Education, ICCSE 2013*, no. Iccse: 37–41. <https://doi.org/10.1109/ICCSE.2013.6553879>.

Xu, Xun. 2012. “From Cloud Computing to Cloud Manufacturing.” *Robotics and Computer-Integrated Manufacturing* 28 (1): 75–86. <https://doi.org/10.1016/j.rcim.2011.07.002>.

Xue, Dongyue, and Eylem Ekici. 2012. “On Reducing Delay and Temporal Starvation of Queue-Length-Based CSMA Algorithms.” *2012 50th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2012*, no. 2: 754–61.
<https://doi.org/10.1109/Allerton.2012.6483294>.

Yassein, Muneer O. Bani, Yaser M. Khamayseh, and Ali M. Hatamleh. 2013. “Intelligent

- Randomize Round Robin for Cloud Computing.” *International Journal of Cloud Applications and Computing* 3 (1): 27–33. <https://doi.org/10.4018/ijcac.2013010103>.
- Zhang, Qi, Lu Cheng, and Raouf Boutaba. 2010. “Cloud Computing: State-of-the-Art and Research Challenges.” *Journal of Internet Services and Applications* 1 (1): 7–18. <https://doi.org/10.1007/s13174-010-0007-6>.
- Manvi, S. S., & Shyam, G. K. (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of network and computer applications*, 41, 424-440.
- Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (2010, April). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In 2010 24th IEEE international conference on advanced information networking and applications (pp. 446-452). IEEE.
- Zhou, A., Wang, S., Cheng, B., Zheng, Z., Yang, F., Chang, R. N., ... & Buyya, R. (2016). Cloud service reliability enhancement via virtual machine placement optimization. *IEEE Transactions on Services Computing*, 10(6), 902-913.