Developing Variance Considered Enhanced Round Robin Algorithm in Cloud Computing Environment

By

Esmael Hassen

A Thesis Submitted to:

Department of Computer Science and Engineering School of Electrical Engineering and Computing

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

Office of Graduate Studies
Adama Science and Technology University

September 2021
Adama, Ethiopia

Developing Variance Considered Enhanced Round Robin Algorithm in Cloud Computing Environment

By

Esmael Hassen

Advisor

Dr.-Ing Frezewd Lemma

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

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

Office of Graduate Studies

Adama Science and Technology University

September 2021

Adama, Ethiopia

Approval

We hereby certify that the recommendation and suggestion given by the "**Developing Variance Considered Enhanced Round Robin Algorithm in Cloud Computing Environment**" by Mr. Esmael Hassen.

_____	_____	_____
Advisor	Signature	Date

We, the undersigned, members of the Board of Reviewers of the proposal open defense by **Mr. Esmael Hassen** have read and evaluated the thesis/dissertation proposal entitled "**Developing Variance Considered Enhanced Round Robin Algorithm in Cloud Computing Environment**" and assessed the understanding of the candidate about the proposed research. This is, therefore, to certify that the thesis/dissertation proposal is accepted and we recommend the implementation of the proposal.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Reviewer 1	Signature	Date
_____	_____	_____
Reviewer 2	Signature	Date

Final approval and acceptance of the thesis proposal is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

Declaration

I declare that this Master Thesis entitled "**Developing Variance Considered Enhanced Round Robin Algorithm in Cloud Computing Environment**" is my work and has not been submitted to any university for a similar purpose. The references used in this proposal are duly recognized by proper citations.

_____	_____	_____
Student Name	Signature	Date

Recommendation

I, the major advisor of this thesis, hereby certifies that I have closely advised the student while developing the thesis and read this Master Thesis entitled "**Developing Variance Considered Enhanced Round Robin Algorithm in Cloud Computing Environment**" prepared under my guidance by Esmael Hassen. Therefore, I recommend the submission of the revised version of the thesis to department following the applicaple procedure .

Dr.-Ing Frezewd Lemma _____ _____

Major Advisor: Signature Date

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude and admiration to my adviser, Dr.-Ing Frezewd Lemma, for his direction, assistance, and encouragement throughout the research study. His advice, guidance, and encouragement have significantly assisted me in overcoming several challenges and finishing this thesis. His words of motivation and advice had a significant impact on me, inspiring me to conduct better research. I would not have been able to finish such a paper if he hadn't consistently accompanied me.

TABLE OF CONTENTS PAGE

LIST OF TABLES

List of Figures

List of Abbreviation and Acronyms

NIST:	National Institute of Standards and Technology
VCRR:	variance considered Round Robin
DCC:	Data Center Controller
IAAS:	Infrastructure as a Service
IDE:	Integrated Development Environment
LBA:	Load Balancing Algorithm
PAAS:	Platform as a Service
RR:	Round Robin
SAAS:	Software as a Service
VM:	Virtual Machine
VMLB:	Virtual Machine Load Balancer
VDI:	Virtual Desktop Infrastructure
EFPRR:	Eight Five Percent Round Robin

Abstract: Today cloud computing is the most elevating innovation due to its revolutionary and momentous power. In cloud computing, task scheduling plays a key role in improving the performance of a cloud system. It plays a vital role in succeeding high-level performance and throughput by having the highest benefit from the resources. Improving job scheduling algorithms will improve the quality of service, resulting in greater cloud computing system sustainability. The task scheduling algorithm is primarily concerned with achieving proficiency and ensuring fairness in the execution of tasks. In this paper, we concentrate on Task scheduling algorithms and on approaches that are meant to generate effective schedules. We have proposed a novel technique which is called variance considered Round Robin (VCRR) task scheduling algorithm. In comparison to several studied algorithms, the experimental findings generated using the Cloud-Analyst tool showed that VCRR greatly improved the average waiting time, turnaround time, context switch, and response time.

Keywords: Task scheduling, Load Balancing, Cloud Computing, Round Robin, Variance

CHAPTER ONE

Introduction

1.1 Background of the Study

The rapid growth of information technology is radically changing the way of world experience in many aspects. The fast spread of computer networks with broadband Internet connectivity, as well as the increasing development of various operating systems, virtualization technologies, and platforms, allows different computing solutions to be used in a variety of fields. [1]. Today cloud computing is the most elevating technology due to its revolutionary and significant power [2]. It is one of the hottest technologies that are being applied in many organizations and institutions. Cloud management system, considered as ground breaking development, has made the provision of cloud services possible.

Virtualization and cloud computing, as well as the growing use of advanced, ultra-portable devices, are all assisting in the enhancement of user experience quality. Internet innovations have accelerated the transition from earlier organizational computing processes characterized by mainframes, client-server structures, and personal computers (PCs), and have restructured the idea of computing into a phenomenon that uses networks all over the world. As a result, IT is increasingly reliant on heterogeneous network networks and protocols. Cloud computing has given birth to some interesting services, such as hosted file servers, on-demand applications, and business continuity solutions [3].

Virtualization is a technology that is the backbone of cloud computing that enables physical computers to share their capabilities by dividing resources among different operating systems [4]. Cloud management systems rely heavily on Internet technologies and virtualization. Cloud management framework, which can be thought of as an evolutionary development, has enabled the provision of cloud services [5]. Cloud provides a good platform and infrastructure to its end-user. Every one of the administrations offered by servers to customers is given by cloud service provider (CSP) which is on a very basic level equivalent to filling in as the Internet service provider (ISP) in the electronic registering[6] The model of running shared desktops in virtual machines hosted on servers is a gripping proposition. In contrast to traditional desktop management strategies, these virtual desktops are simple to

maintain, upgrade, and update and users may access sensitive data from a variety of devices in various places without ever leaving the data center [7].

In cloud computing, Virtualization systems are being used by various organizations to reduce the overall cost of IT equipment and administration, as well as to simplify resource and technology management [9]. Running shared desktops in virtual machines on servers is an enticing concept. Unlike conventional desktop management techniques, virtual desktops are simple to maintain, upgrade, and update, and without ever leaving the data center, users can access sensitive data from a variety of devices in various places. However, implementing VDI has been a difficult task and complex responsibility for large enterprises due to demanding requirements for high-end server and storage hardware [10]. Several vendors are now combining virtualization and cloud computing to take the technology to the next level. [3]. With the emergence of cloud computing and virtualization technologies, many remote desktop solutions are used to access the desktop environment running on remote Virtual Machines (VMs) offered by commercial Infrastructure as a Service (IaaS) providers such as Amazon, Rackspace, and many others [11].

In virtual desktops, everyone is served the same fast, secure, high-quality computing experience. The use of desktop virtualization in cloud computing has been embraced by many leading IT companies such as Microsoft, Google, Amazon, HP, VMware, and IBM [12]. Voluminous establishments use VDI technologies from vendors like Citrix and VMware to push virtual desktops out across their enterprises [3].

Cloud Computing has a great potential to provide maximum storage and computational power at a low and affordable cost. It builds on virtualization and distributed computing developments to enable cost-effective computing resource use, with a focus on resource scalability and on-demand facilities [2]. Cloud computing virtualization offers a viable solution for resource consolidation and management simplification.

However, cloud computing has several critical issues, such as security, load adjusting, adaptation to non-critical failure, and so forth.[6] There are numerous existing issues in cloud computing. However, the key amongst them is task scheduling. Task scheduling is the process of organizing incoming requests

(tasks) in a specific order to maximize the use of available resources [44]. One of the most critical aspects of a cloud computing system is scheduling. Task scheduling is one of the most important study topics in a cloud computing environment. With the growing number of cloud customers, service providers must focus on providing adequate access to remote resources while also maximizing profits. The process of determining which tasks will be mapped to which resources are known as resource scheduling.

Task scheduling is a strategy that uses an efficient algorithm to map clients' tasks to available and appropriate virtualized resources [10]. Task scheduling is the most important since it is the primary determinant of other performance metrics including availability, scalability, and power consumption by enabling all waiting processes in a computing system to make efficient and appropriate use of computing resources. Task Scheduling is the process of allocating resources to a specific task to complete it in the shortest amount of time possible. Task Scheduling aids in maximizing the use of available resources. Task scheduling plays a vital role to accelerate networks and resources by providing a maximum throughput with minimum response time [10]. The goal of Task scheduling algorithms is to enable resources to execute tasks with minimum waiting and execution time.

Due to the aforementioned challenges of cloud computing, it is valuable to research this topic to improve the existing task scheduling algorithms in cloud computing data centers. Ultimately this will create the optimal solution for desktop virtualization providers and end-users. This paper focuses on developing enhanced round-robin task scheduling algorithms in cloud management systems and virtual desktop infrastructure for better Quality of Service.

The remainder of this thesis is structured in the following manner. In chapter 2, the literature review and related works are presented. In chapter 3, the problem statement is explained, while the proposed technique is explained in detail in

chapter 4. Chapter 5 covers the evaluation and discussion of our proposed work. Finally, and future work and the conclusion are comprised in the last section.

1.2 Statement of the Problem

The massive growth of cloud computing would bring an extra load on the server. As a result, it will lead to degrading the performance of networks [6]. This problem persists and propagates with the cumulative customer base in the cloud system [5]. Managing the customer demand creates the challenges of on-demand resource allocation [2] Especially when organizations and institutions dumped local computers and start using virtualized cloud desktops, an enormous number of end-users will access the cloud server. Ultimately this will create a big burden on the cloud server. In this case, the big challenge will be task scheduling and load balancing. The existing algorithms for task scheduling do have not enough capability to handle a massive number of requests [34].

Due to the diverse and varied tactics that can be used, ranging from virtual machine locations to collaboration with other clouds, cloud data center administration is a critical issue. [5]. So, task scheduling is the main challenge in a cloud environment. The solution to this problem is to ensure a mechanism that enables the server to perform an execution with less waiting and execution time which enhances overall system efficiency, lowering response time, and reducing total system cost. Task scheduling is important for essential operations in cloud virtual environments [44]. But still, there is difficulty in finding an efficient task scheduling algorithm with minimum response time and optimum processing time, throughput and resource utilization [10].

A cloud-based virtual desktop infrastructure (VDI) is a computer approach in which an end user's system can virtually access all of their important files and data while alienated from the actual IT infrastructures. However, issues of Task scheduling discourage end users from cloud management systems. The capacity of data storage grows rapidly in an open environment. The massive growth of cloud computing would create over burden on servers. Because cloud

computing serves millions of users at once, it must be able to meet all requests with excellent performance and a guarantee of service quality.

As a result, to answer these requests properly and efficiently, we must build an appropriate job scheduling algorithm. Because cloud performance is largely dependent on task scheduling, it is one of the most crucial concerns in the cloud computing environment. There are several types of scheduling algorithms, including static scheduling methods that are appropriate for small or medium-scale cloud computing and dynamic scheduling techniques that are appropriate for large-scale cloud computing. Task scheduling is one of the most important topics of research in the field of cloud computing today, as it aids not only in optimal resource usage but also in avoiding deadlock inside the cloud environment [34]. One of the most critical aspects of an operating system is scheduling. It enables all waiting processes in a computing system to make efficient and appropriate use of computing resources. Due to the presence of a large number of computing resources that are all integrated into a single computing network to be able to behave and operate as a single computing resource of higher capabilities and capacity, scheduling algorithms become even more crucial in the cloud computing environment. So, Task scheduling is the main challenge in a cloud environment. It is very essential for proper utilization of resources as well as to improve the performance of the system. Task scheduling is important for essential operations in cloud virtual environments.

The popularity of cloud computing platforms has risen dramatically in recent years. Task scheduling is one of the most difficult aspects of developing cloud computing systems. It is very important to obtain high-level performance and remarkable throughput by utilizing available resources effectively. As a result, improving job scheduling algorithms will improve QoS, resulting in greater cloud computing system sustainability [45].

Round Robin is the first of the most well-known scheduling algorithms. It was created specifically for time-sharing systems and is one of the simplest, fairest,

and most extensively used scheduling algorithms [17]. For this reason, different researchers and investigators work on improving the Round Robin algorithm by concentrating on choosing a suitable quantum time and they proposed various techniques [17]. But in the case of the RR algorithm, no process waits for the completion of other processes as execution is done based on time-sharing. However, round-robin is most widely used, the major drawback is it uses static Time Quantum. To tackle this, Matarneh has come up with the model of dynamic TQ for better results of TAT, WT, and NCS [36]. It is a basic challenge to calculate the best TQ for better results of TAT, WT, and NCS. To make a solution for this challenge many researchers have proposed different algorithms.

To overcome the existing problem and make improvements to the RR scheduling algorithm we proposed a new approach by creating a new novel technique to choose the best quantum time. Appropriate task scheduling can keep minimum resource consumption which will achieve further reduction of energy consumption and carbon emission rate. In general, task scheduling mechanisms in computer environments need to improve in terms of managing the heterogeneity of their environments to become fully on-demand techniques, lowering related overhead, improving service response time, and enhancing performance[18]. On the other hand the author in [19] has described many technical challenges that need to be tackled. Virtual machine migration, server consolidation, fault tolerance, high availability, and scalability are from those challenges. But the major challenge and the central issue is task scheduling. So, it is worthy to research task scheduling algorithms for further improvement.

The main contribution of this research is to provide a new technique that focuses on the drawbacks of the typical RR algorithm. The proposed model improves the functioning of the typical RR algorithm in the cloud computing environment by reducing waiting time, average turnaround time, and response time.

1.3 Research Questions

In this research, the following research questions are intended to be addressed.

- ✓ How we can design variance considered a round-robin (VCRR) scheduling algorithm?
- ✓ How to develop a task scheduling algorithm that has a fast response time while still maximizing resource utilization?
- ✓ How to test the proposed algorithm using scheduling algorithms performance metrics such as average turnaround time, average waiting time and context switch?

1.4 The objective of the Study

1.4.1 General Objective

The main goal of this study is to design and develop a better and more advanced Round Robin Task-Scheduling algorithm for cloud computing.

1.4.2 Specific Objectives

The study will aim to discuss the following basic goals to achieve the above-mentioned key goal:

- ✓ Reviewing related works on task scheduling techniques in cloud computing.
- ✓ Identifying drawbacks and limitations in the existing task scheduling algorithms
- ✓ Designing variance considered a round-robin (VCRR) scheduling algorithm.
- ✓ Developing a task scheduling algorithm that has a fast response time while still maximizing resource utilization.
- ✓ Testing the proposed algorithm using scheduling algorithms performance metrics such as average turnaround time, average waiting time and context switch.

1.5 Scope and Limitation of Study

In this study, we proposed a modified task scheduling algorithm based on the best available current widely used algorithms, to overcome the algorithms' limitations in terms of computational time, response time, and resource utilization. Other cloud computing issues

aren't included in this research. A Modified Advanced Round Robin task scheduling algorithm is proposed in this paper. The proposed algorithm's output was evaluated in a simulator rather than in real-world experiments.

1.6 Significance and beneficiaries of the Study

The proposed algorithm will be used for better resource utilization, computational and response time minimization. Because the optimum task scheduling algorithm will help to regulate the entire load of the system, cloud providers can efficiently handle a cloud resource and Cost-effectiveness will improve. The proposed algorithm can be used to improve the overall performance of the cloud environment. In addition, the research may be used as a starting point or a guide for additional future research and studies on new task scheduling and task scheduling algorithm strategies.

CHAPTER TWO

Literature Review

2.1. Overview

This chapter presents the most relevant concepts related to study work. It offers a detailed overview of cloud computing, including its fundamental features, service model, and deployment models. Furthermore, it addresses cloud computing issues, especially task scheduling issues in cloud computing using existing task scheduling algorithms.

2.2. Cloud Computing

Cloud Computing is an emerging network technology that has seen rapid advancements in communication technology by delivering service to consumers with a variety of needs using online computing services.[14] Cloud computing is a network-based computing model that allows users to share resources as services that are billed on a pay-as-you-go (PAYG) basis. Cloud computing is a cost-effective service provisioning model that makes IT management simpler and more sensitive to changing business needs [29].

According to the National Institute of Standards and Technology (NIST), the definition of cloud computing is [3].

- ✓ As a demand-driven service, providing information to users.
- ✓ Access to computer network management and software is available from anywhere and at any time.
- ✓ Numerous users can share data. Data can be shared by a large number of people. The cost is determined by the vendor's location and user consumption time, as well as the vendor's sustainability and the vendor's environment.
- ✓ Resources will be allocated according to changes in demand for services, and they can be processed quickly.
- ✓ The cost is determined by location and usage time, as well as the vendor's environment.

✓ Users do not need to concern themselves with assigning resources to the service. The service is measured by the number of users or the length of time it is available.

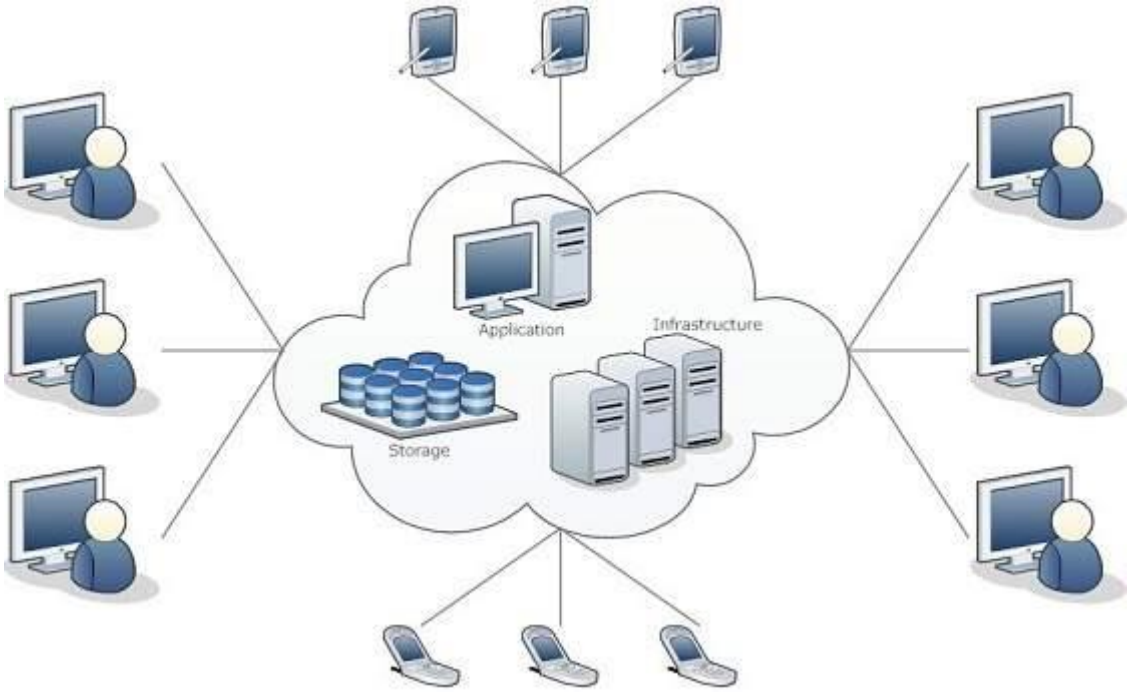


Fig 1: Cloud computing over view-source: https://www.tutorialspoint.com

2.2.1. Cloud Computing Characteristics

According to the National Institute of Standards and Technology (NIST) [31], cloud computing has the following fundamental characteristics:

- **On-demand self-service work**: Operators of cloud computing services may access their computing service without requiring assistance from others or interacting with cloud service providers.
- **Admission to a wide network:** Cloud services are available over the internet and can be accessed via a variety of small and large client devices such as cell phones, tablets, computers, and Personalized Digital Assistants.
- **Rapid elasticity**: A cloud service provider may scale up or down their ability to meet the needs of their customers without compromising the quality or the integrity of their services.
- **Measured service**: By using a metering capability at a specific point of abstraction, the overall cloud architecture automatically manages and optimizes resource use like storage, bandwidth, currently active user accounts, and processing ability. For both the

service provider and the customer of the used service, resource use may be tracked, handled, and recorded for the intended uses, providing transparency and accountability

- **Resource Pooling:** Using the benefit of a multi-tenant model, cloud service providers may offer their services to a large number of clients concurrently. As a result, there is no direct line between geographical area and site dependence, and the client is frequently unaware of the precise position of the service being offered. Therefore, the user may be required to specify the location (e.g. continent, nation, state, or exact data center) at a high degree of abstraction.

2.2.2. Cloud Service model

Cloud computing has three basic models which are Platform as a Service (PaaS), Infrastructure or Hardware as a Service (IaaS/Haas), and Software as a Service (SaaS).

Software as a service (SaaS): All applications are delivered to the customer in this service delivery model, and the consumer is not responsible for the underlying infrastructure. Users can use web browser interfaces to access services. Google, Salesforce, and Microsoft are some of the companies that offer SaaS. It was the first significant cloud industry, and it is still expanding rapidly. SaaS is a delivery model that uses the internet to distribute applications that are operated by a third-party provider. While some SaaS applications require plugins, most SaaS applications can be run directly from a web browser without the need for downloads or installations [32].

Platform as a service (PaaS): in this service delivery model the customer can gain access to services that enable them to build applications and they are not responsible for the application's download and installation. They deploy all the applications onto the cloud infrastructure. Also, the consumer does not control networks, servers, operating systems, or storage. The consumer controls all applications which they deploy.

Google’s App Engine, Force.com, are some of the common examples of PaaS. Cloud platform services, also known as Platform as a Service (PaaS), are used

to build applications and provide cloud components to software packages. Developers will gain a framework with PaaS that they will rely on to develop or customize applications. PaaS makes application development, testing, and deployment fast, easy, and effective. Enterprise operations, or a third-party vendor, can handle OSes, virtualization, servers, storage, networking, and the PaaS software package itself with this technology [32]

Infrastructure as a service (IaaS): In this service, the consumer does not manage or control the underlying cloud infrastructure and setup. In this service delivery model, the end-user is responsible to control operating systems, storage, and all applications which they deployed. Service providers control storing and processing capacity.

Some common examples are Amazon, GoGrid, and 3 Tera. It is a self-service model for accessing, controlling, and maintaining remote data center infrastructures such as computer storage, networking, and networking services in the cloud (e.g. firewalls). Instead of purchasing hardware outright, users can purchase IaaS-supported consumption, which works similarly to power or other utility billing [32]

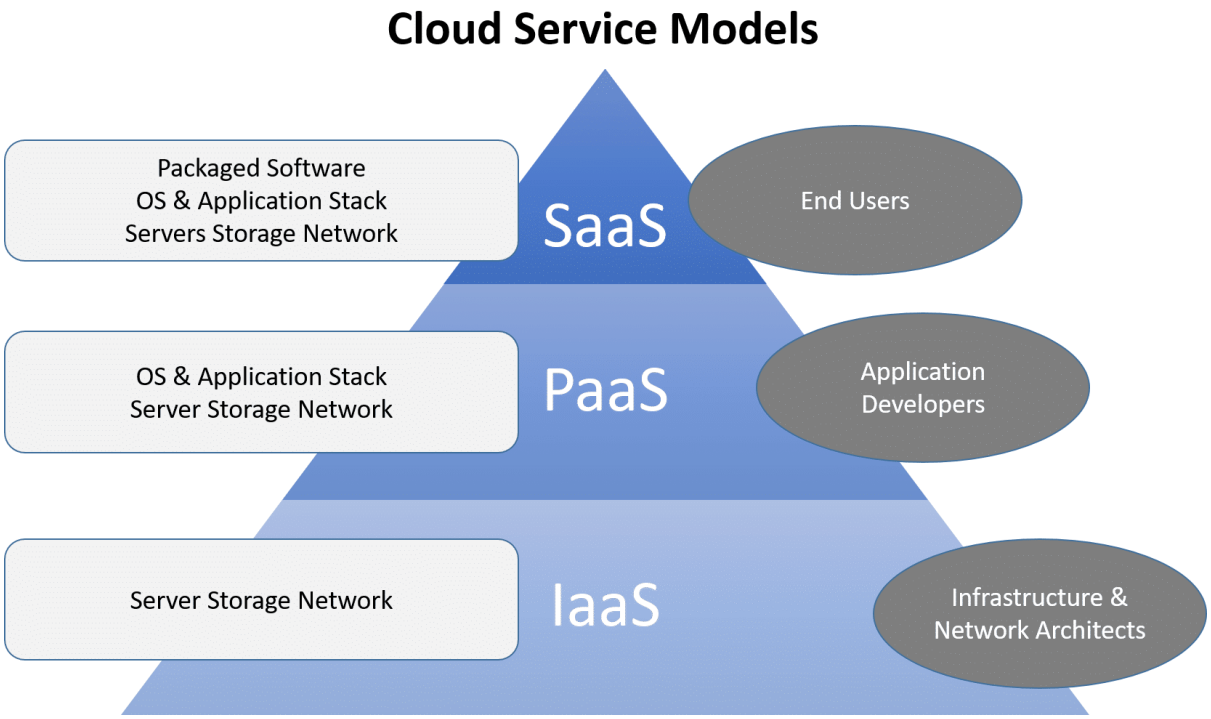


Figure 1 cloud service models

Source: https://www.uniprint.net/en/7-types-cloud-computing-structures/

2.2.3 Cloud deployment model

2.2.3.

There are four types of cloud computing deployment models [6]:

- **Private cloud**: this cloud computing is managed and controlled by a single association or by a specific organization, and this institute utilizes its infrastructure as well.[6]
- **Community cloud:** this cloud is overseen by a few associations and supports a particular network and has shared the applications or services.
- **Public cloud:** this model possesses and is overseen by a huge cloud service provider (CSP). All customers who need to use resources on a participation premise or on a membership premise [6].
- **Hybrid cloud:** it is the combination of at least two of the above models, i.e. public, private and community model [6].

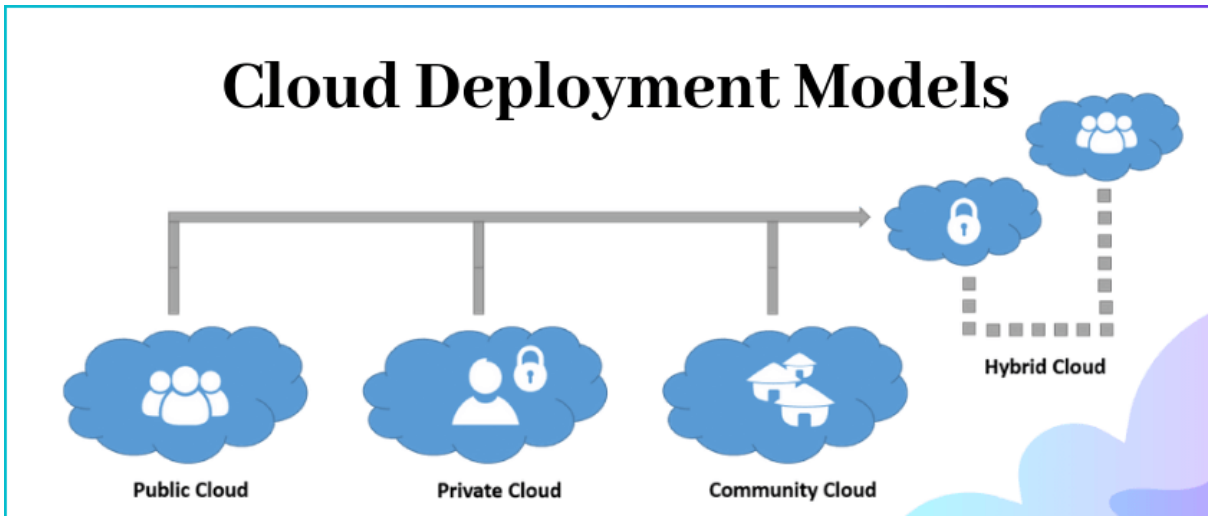


Figure 2 cloud deployment models

Source: https://medium.com/

2.2.4 Advantages of cloud computing

Cloud Computing's present high levels of interest, as well as the positive coverage it has received, are largely fueled by its perceived benefits. According to [36], the ideal Cloud Computing infrastructure would include the following characteristics

- ✓ **Self-recovery:** In the event of a failure, a hot backup instance of the program will be ready to take over without halting any process. The system creates a fresh backup when one of the backups becomes the primary.
- ✓ **SLA-driven:** Service-level agreements, which establish policies such as how quickly responses to requests must be given, are used to dynamically govern the system.
- ✓ **Multi-tenancy:** The solution is designed to allow several clients to share infrastructure while maintaining mutual transparency and ensuring privacy and security.
- ✓ **Service-oriented:** The technology enables the creation of applications from a collection of discrete, reusable, loosely connected services. Changes to a service, or its failure, will not affect other services.

- ✓ **Virtualized:** The underlying hardware is isolated from the applications. Grid Computing might be used by one application, and numerous apps could run on a single machine.
- ✓ **Easily and Efficiently Scalable:** With increasing demand, the system will scale predictably and efficiently.
- ✓ **Data-Oriented:** The administration of data: its distribution, partitioning, security, and synchronization utilizing technologies like Amazon's and in-memory data grids are the key to many cloud features.

A properly executed Cloud Computing system should provide a customer with some or all of the following benefits, according to Salesforce.com which is a leading Cloud Computing vendor: [36]

- ❖ **Proven Web-services integration:** Cloud Computing technology makes integrating with other enterprise software much simpler.
- ❖ **World-class service delivery**: Cloud Computing providers provide far more scalability, total disaster recovery, and remarkable availability figures.
- ❖ **No need to install hardware or software**: Cloud Computing takes substantially less cash to start-up and running.
- ❖ **Lower-risk deployment:** No more waiting months or years for anyone to log into your new solution and investing millions of dollars. Even with considerable customization or integration, Cloud Computing technology applications come to life in a couple of weeks or months.
- ❖ **Support for deep modifications**: The Cloud Computing architecture not only provides for deep customization and application setup, but it also keeps those customizations even during upgrades and when needs change, freeing up IT resources within the organization.
- ❖ **Empowered business users**: Cloud computing technology enables business users to customize and generate reports on the fly, reducing the amount of time.

- ❖ **Hundreds of pre-built applications** and application exchange capabilities are either pre-integrated into bigger, off-the-shelf applications or available for simple integration to construct new applications.

2.3. Task scheduling Algorithms Metrics

Task scheduling is the process of organizing incoming requests (tasks) in a specific order to maximize the use of available resources [44]. One of the most critical aspects of a cloud computing system is scheduling. Task scheduling is one of the most important study topics in a cloud computing environment. With the growing number of cloud customers, service providers must focus on providing adequate access to remote resources while also maximizing profits.

Scheduling of resources is the process of determining which tasks will be assi gned to which resources. Task scheduling is the process of determining which tasks will be assigned to which resources. Task scheduling is a strategy that uses an efficient algorithm to map clients' tasks to available and appropriate virtualized resources [10] Task scheduling algorithms are applied to increase the overall cloud environment working capability. Cost, scalability, flexibility, and its executing flow are some major factors that determine the capacity and efficiency of an algorithm. In cloud computing, many types of Task scheduling algorithms have been designed which the main goal is to achieve maximum throughput with a minimum response time. Principally, Task scheduling algorithms are classified into two: Static task scheduling algorithm and Dynamic task scheduling algorithm. The following parameters are used to evaluate the efficiency and effectiveness of the task scheduling algorithm in the cloud system [15].

- ✓ **CPU Utilization:** The amount of time that the CPU is in a busy state – not idle.
- ✓ **Throughput:** Tasks accomplished per time unit.
- ✓ **Waiting Time:** The total time spent in the ready queue during the life cycle of the process. Time spent waiting for I/O, is not included in the waiting time.

- ✓ **Turnaround time**: a process is the duration of a process from when it enters the ready state to when it exits the running state.
- ✓ **Response time** – The elapsed time between the demand placed and the beginning of a response after completion of the task.
- ✓ **Scalability** – Ability of computer applications to continue their function effectively even when their size and topography change.
- ✓ **Fault tolerance**: A system that is designed to withstand failure and continue to function despite it.
- ✓ **Overhead**: Term used to describe the amount of time taken by the system to complete an installation, operation, or any transaction inside it.
- ✓ **Power Consumption** – Information technology consumes tremendous power and involves high energy costs. Efficient power management is vital for the success of an IT environment such as cloud computing systems.
- ✓ **Complexity** – Making the entire system difficult. With increasing users associating with the cloud and its properties, the complexity of the system increases.
- ✓ **Fairness** – Indicates that each user has an equal response time and all get their jobs completed within approximately the simultaneously.
- ✓ **Performance** – It is the speed and accuracy at which the jobs are completed and is measured against the preset standards.

2.4. Virtualization and Virtual Desktop Infrastructure

Virtualization refers to something that doesn't exist in the physical world but gives service as it is real. Virtualization is the software implementation of a computer that can run several programs in the same way that a physical machine can. Users can access the cloud's various applications and services via virtualization. This is potentially the most important aspect of the cloud. Many forms of virtualization exist in a cloud-based setting.

[27] Defines virtualization as "a means of abstracting a computer's physical resources into virtual ones with the help of specialized software. Abstraction layers allow several virtual machines [VMs] to be created on a single physical machine. Virtualization affords encapsulation of processing capabilities of IT

resources into virtual platforms and executes these virtual platforms on a host machine in an isolated way. Many organizations are exploiting a variety of virtual platforms in response to the challenges organizations are experiencing with regards to computing [28].

There are two types of Virtualizations:

- ✓ **Full Virtualization**: When a complete machine is mounted on another machine, it is known as full virtualization. The virtual machine performs all of the tasks that the original machine does. The user can only use the virtual machine when a physical machine is unavailable.[7]

- ✓ **Para virtualization**: When hardware enables multiple operating systems to operate on a single computer, this is referred to as para virtualization. It also ensures that device resources including memory and processor are used efficiently.[7]

VDI is a mechanism for hosting a desktop operating system on a server in the form of a virtual machine (VM). A user at an endpoint workstation uses a remote display protocol to connect to the VM over the network, allowing the virtualized desktop to be rendered locally [5]. The idea of desktop virtualization is to separate a logical operating system (OS) instance from the client that accesses it. It is a method of hosting a desktop operating system inside a virtual machine (VM) running on a centralized server.

2.5. **Task Scheduling Classification**

Scheduling in cloud computing is generally divided into several categories [13]. The first type of classification is based on Task. Static and dynamic task scheduling are the two types of task scheduling. In static scheduling, tasks arrive at the processor at the same time and are assigned to the processor's available resources; scheduling decisions are made before tasks are distributed. After a job is completed, the processing time is updated; this type of task-based scheduling is typically used for periodic tasks. The number of tasks, machine location, and resource allocation is not fixed in dynamic scheduling. In dynamic scheduling before submission, the task arrival timings are unknown. Dynamic scheduling is further divided into two types: batch mode and online mode. Tasks are queued, collected into a set, and scheduled

after a predetermined amount of time in batch mode. Tasks are scheduled in online mode when they arrive in the system.

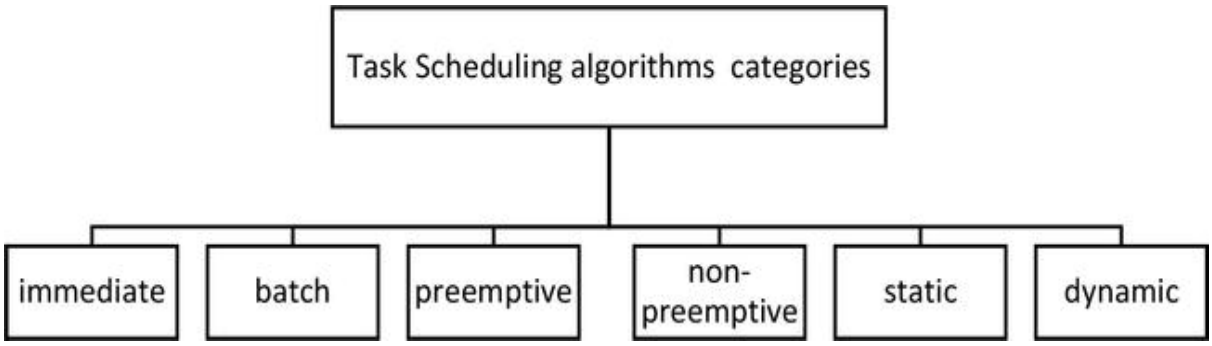


Figure 3 categories of Task scheduling algorithms [44]

The second group is divided into batch, interactive, and real-time systems based on the various metrics utilized in cloud computing settings. The turnaround time and throughput of a batch system can be calculated. An interactive system may be used to calculate response time and fairness, and the system's deadline can be measured in real-time.

2.6. **Task scheduling models**

Task scheduling is used in a cloud environment to effectively schedule tasks at the cloud platform and allocate specific tasks to appropriate resources to maximize resource utilization, improve performance and QoS at the same time, and achieve high efficiency and economy at execution time. Various scheduling models exist, based on the scheduling criterion. Task scheduling algorithms in the cloud computing sector are classified into the following primary groups based on a simple taxonomy.

Static Scheduling model: From the perspective of the scheduler, because all available resources and jobs are known, and each resource is assigned to a job just once, it is a simpler strategy. 16].

Dynamic Scheduling model: This technique is more flexible than static scheduling, but it has a higher execution cost [16]. It dynamically performs tasks by allocating resources at runtime.

- **Hybrid Scheduling Model:** This is a hybrid scheduling model that combines the prior two scheduling strategies. This method is employed in virtual machines.
- **Distributed Scheduling Model**: This scheduling model is better suitable for cloud platforms; it is a realistic method in which job requests are received on a real-time basis. When compared to centralize scheduling, this strategy is more brittle, but it is more realistic when scheduling in a distributed computing.
- **Centralized Scheduling Model:** This strategy is more adoptable, simple to deploy, and manageable. However, because this method lacks scalability, it is insufficient for large-scale clusters.
- **Cooperative Scheduling Model:** This is a scheduling strategy in which different schedulers work together in a group to accomplish their duties in a common system according to stated procedures.

2.7. Task scheduling Algorithms

Different scheduling algorithms have been proposed in cloud computing, with the main goal of achieving high throughput and minimal response time. In general, there are two types of task scheduling algorithms:

A. Static scheduling algorithm
B. Dynamic scheduling algorithm

2.8. Static scheduling algorithm

The static scheduling algorithm is a type of scheduling algorithm that assigns the task to the CPU based on a set of fixed rules, such as input work-load. The load is independent of the system's current state, but it does necessarily require knowledge of the program and its resources. First come, first served (FCFS), Round Robin algorithm, Shortest Job First, Min-Min, Max-Min are the most well-known different types of static scheduling techniques.

First come, first served (FCFS): The term "First Come, First Serve," sometimes known as "First in, First Out," refers to the sequence in which jobs are completed. The tasks are executed as per the order of task arriving time. The convoy effect, which occurs when a job with a large amount of workload is queued, may be exacerbated by the FCFS algorithm. All of the jobs behind it in line will have to wait a long time for the long job to finish in this scenario.

Round Robin Algorithm: It's one of the most basic scheduling algorithms that work with time slices. Time is divided into several slices in this case, and each node is assigned a specific time interval. Each node is assigned a quantum, within which it must carry out its actions. If the user's request is completed inside the time limit, the user should not wait; otherwise, the user will have to wait for the next available slot. The round-robin scheduling algorithm's fundamental idea is the quantization of computing time [34]. The dividing of available computational time into smaller chunks of specified sizes is referred to as quantization. As a result, the scheduler traverses all accessible processes sequentially, processing each one for a set length of time. As new processes arrive, they are automatically added to the rear of the queue [44]. The effectiveness of Round Robin is due to the way the time quantum is determined. The longer the quantum, the more probable the behavior of First Come First Serve will be replicated (FCFS). The shorter it is, the less effective it becomes, as more time is wasted switching contexts.

Shortest Job First Task Scheduling Algorithm: Shortest Job First (SJF) choices the process with the minimum execution time. The highest priority is assigned to the task s with minimum execution time and positioned first in the queue while the lowest priority is assigned to the process with the largest execution time. It might be preemptive or non-preemptive. A pre-emptive SJF algorithmic rule can halt the running method, but a non-pre-emptive SJF algorithmic rule can allow the running method to complete its half.

Min-Min Task scheduling algorithm: The Min-Min algorithm's working principle is to map each task to resources in such a way that they can complete

the task in the shortest amount of time. It calculates how long each job will take to execute and complete on each available resource. The Min-Min algorithm is divided into two parts. It calculates the task with the shortest execution time in the first phase. In the second phase, the task with the shortest execution time is chosen out of all the tasks. The algorithm then assigns the work to the resource that will complete it in the shortest amount of time. The process is repeated until all of the tasks have been scheduled.

Max-Min Task Scheduling Algorithm: The Max-min algorithm is quite similar to the Min-min method in terms of how it operates. The distinguishing feature is that the word "min" is substituted with "max," implying that the job with the earliest completion time is assigned to the related resource. Larger tasks take precedence over lesser tasks in this case.

2.9. Task Scheduling Level

There are two types of scheduling algorithms in the cloud computing environment [37]:

- The first level is at the host level, where a set of policies govern how VMs are distributed within the host in the cloud.
- Second level: at the VM level, a set of policies for assigning tasks to VMs.

In this paper we chose task scheduling algorithms as a research field because it is the biggest challenge in cloud computing and the main factor that controls performance criteria such as (execution time, waiting time, response time, bandwidth, network, services cost) for all tasks as well as controlling other factors that can affect performance.

2.10. Task scheduling Layer in cloud computing

In cloud computing, the job scheduling system has three layers [37].

- The first task level consists of a series of tasks (Cloudlets) that are sent by cloud users and must be completed.

- The second scheduling level is in charge of assigning tasks to appropriate resources to maximize resource utilization while keeping the turnaround time to the smallest. The makespan is the total time it takes to complete all tasks from start to finish.
- The third level of virtual machines (VMs) is a collection of virtual machines (VMs) that are utilized to complete activities.

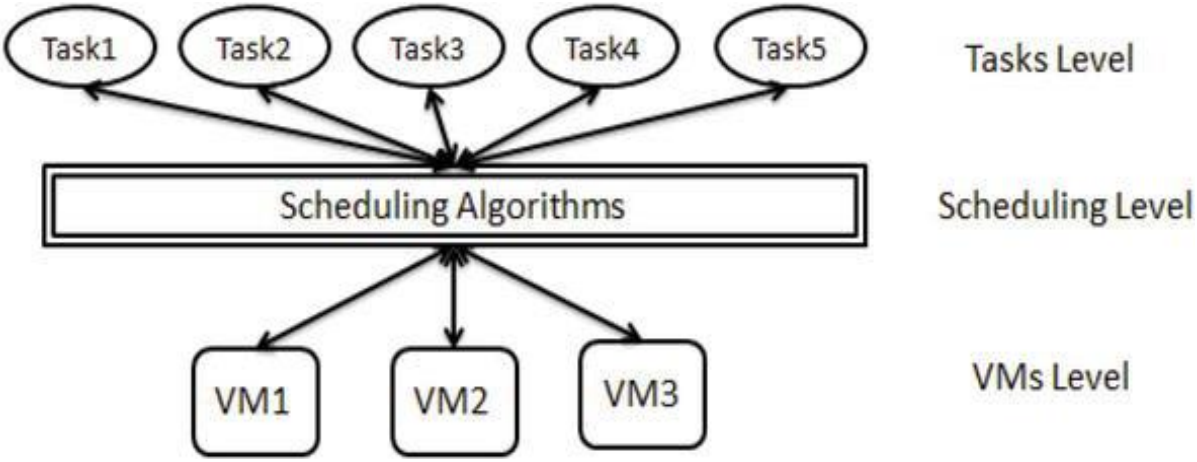


Figure 4 Task scheduling model

Source: https://www.intechopen.com/media/chapter/71902/media/F2.png

2.11. Benefits of task scheduling algorithms

- Improving the level of fairness in all tasks.
- system's throughput improvement
- Improving load distribution among VM
- Improving the level of fairness in all tasks.
- Control cloud computing performance and quality of service.
- Manage the memory and the processor.
- The best scheduling algorithms make the best use of resources while reducing the total job execution time.
- Increasing the number of tasks completed successfully.
- Task scheduling in a real-time system.

2.12. Characteristics of task scheduling algorithms

In a cloud environment, the system's performance has a significant impact on the revenue of the service provider, so job scheduling becomes a critical liability. Because scheduling is an NP-hard problem, there is always room for improvement. The following are the elements of a good scheduling framework [43]:

A. **Energy consumption:** It refers to how well the system's resources are being utilized. A smart data center scheduling algorithm should prioritize energy efficiency and balancing the load.
B. **QoS Parameters**: The user sets these parameters, which include execution time, cost, and so forth.
C. **Security features**: It should fulfill the security needs.
D. **Fair resource allocation:** The fairness with which resources are allocated has an impact on overall performance.
E. **Resource utilization:** The entire revenue is determined by how well resources are utilized. A smart scheduling algorithm should prioritize resource consumption.

2.13. Related work

A lot of research has been conducted in the area of cloud computing specifically in resource management, task scheduling, resource provisioning, energy management, and load balancing. However, task scheduling has been a central issue among researchers because of its core role in cloud computing between the Cloud Service Provider and Cloud Service end-users [14].

Among those researchers, Alaa's finding has improved the original round-robin scheduling algorithm using his proposed analytical model. He proposed a very admirable method to improve the Round Robin scheduling Algorithm. He put into consideration the type of task (heavy or light) and their arrival time in the selection of quantum time (TQ). His experiment result shows excellent improvement in TAT, WT, and NCS. It is a very great algorithm for few processes that are arrived at either at the same time or at a variable time. But

the problem with his proposed approach is when a large number of processes arrive at the same time the algorithm tends to work like SJF (shortest job first) which leads to starvation for heavy tasks [35]. The proposed analytical model for sorting the list in the ready queue update every time new processes arrived. This gives priority to processes that have small BT, as a result, long processes which have high BT will wait for a longer period in the RQ. This leads to job starvation [35]. His proposed Approach has the following two major drawbacks: Updating the queue before the completion of the execution cycle which results from job starvation for long processes. The quantum time chosen based on the middle task would not be the best choice for the processes when the number of processes becomes very large.

P. Pradhan has been proposed a modified RR algorithm in which the quantum in the first round is equal to the first task's burst duration. The quantum is then updated to the average burst times for all tasks in the ready queue, including the new one if the ready queue has received a new task. The MATLAB results indicate a substantial increase in reducing turnaround and waiting times [20].

] F S. Elmougy and M. Joundy have suggested a new strategy by combining the SJF algorithm and the Round Robin algorithm, for reducing task waiting time. The new technique's key principle is to calculate the time quantum dynamically for each mission and each round. Furthermore, the median was used as a criterion for dividing the task-ready queue into short and long tasks. The CloudSim-based experimental result indicates that the novel technique finds an adequate solution to the problem of starvation in long tasks. Furthermore, as compared to SJF, RR, and Time Slice Priority Based RR, it has demonstrated good performance in terms of reducing waiting time and response time [21].

P. Sangwan proposed a new modification to the RR algorithm in his paper. The processes have been inserted in the ready queue in the order in which they were received. The quantum time was measured using the mean of the processes' burst times. The evaluation of this method was carried out using the Java programming language, and the results show that it performs well in terms of waiting time and turnaround time [22].

Ravichandran, S., and E. R. Naganathan published another analysis that used a genetic algorithm to solve the problem of task arrival uncertainty in the cloud. As a result of this problem, task binding to virtual machines becomes tedious. Dynamic scheduling was proposed as a solution to this problem, in which users' tasks are queued and the scheduler's responsibility is to sort them based on computation and memory usage; then, GA is used to select each task and find the best fit for allocating a task to available virtual machines, resulting in global optimization [23].

S. Banerjee and A. Chowdhury have present an efficient scheduling strategy in their paper, which improves the efficiency of the RR algorithm and the Robin Cloudlet Scheduling Algorithm. The tasks will be allocated to a suitable VM in this method. As a result, each VM will have its local queue. The tasks on these LQ will be scheduled using RR with the optimum quantum, which will be determined based on the median value of average task execution time and the task with the longest execution time. The assessment found that resource utilization, average waiting time, and average turnaround time are all improved [24].

The eighty-five percentile RR method was developed by the researchers as a new strategy (EFPRR). It is based on calculating a TQ using 85 percent of the tasks' burst time. The operations are first ordered in order of burst time, and then the TQ is determined by multiplying the average duration of all jobs by the 85 percent constant (eighty-five percent average). The remaining BT for the tasks is then checked. The ongoing tasks are conducted until they are completed if it is less than or equal to the TQ. If not, the tasks are moved to the end of the ready queue. The EFPRR was tested in numerous experiments, and the results showed that the suggested method was effective at reducing the average [33].

Balharith, T. and Alhaidari F. have conducted a study to improve the RR algorithm and published their findings. We discovered that some publications advocated enhancing the performance of the RR algorithm by computing a dynamic quantum time, which may be dynamic in each round or for each task, while others proposed improving the performance of the RR algorithm by considering a constant quantum time.

A new RR algorithm was developed by B. Fataniya and M. Patel based on the MRRA and SRBRR algorithms. It has been proposed to improve cloud-computing resource utilization. It's a dynamic RR in which the quantum time varies from round to round and is equal to the sum of the mean and median of processes divided by two. According to the results of the experiment, the proposed algorithm outperforms in terms of the number of context shifts, average waiting time, and average turnaround time [25].

S. Banerjee developed and tested an improved RR algorithm in [26]. Initially, cloud users submitted their tasks to a global queue. These tasks are distributed around the VMs using FCFS. As a result, each VM has its task queue. The cloudlets will then be scheduled using the RR policy, and the time quantum will be determined based on how long these tasks take to complete. In comparison to RR, the CloudSim results show that the IRRCSA had a good performance. It reduced the TAT, WT, and number of context switches in particular.

Name of algorithm	TQ finding technique	Strong side	Comments
IVRR algorithm [35]	He put into consideration the type of task (heavy or light) and their arrival time in the selection of quantum time (TQ).	His experiment result shows excellent improvement in TAT, WT, and NCS	evaluation of the Cost of the task in the analytical model takes too much time and space
Hybride SJF and Round Robin algorithm [21]	Combination of SJF and RR TQ finding method	better performance than SJF and RR separately	sorting cost is very high
EFPRR algorithm [33]	It is based on calculating a TQ using 85 percent of the tasks' burst time	it is easy to implement and improve the original RR in terms of ATA	finding TQ every time has a high cost

IRR algorithm [22]	Quantum time is equal to the sum of the mean and median of processes divided by two.	Give the best result in terms of AWT, TAT, and Number of CS.	the computational cost for mean and median is high
DPTQRR [46]	Time quantum has been evaluated dynamically based on the maximum difference among the differences of adjacent consecutive task burst times	Better in terms of ATAT and number of context switch	In some cases, it may lead to starvation
Modified RR algorithm [20]	The quantum is then updated to the average burst times for all tasks in the ready queue.	It is simple to implement and enhance the original RR performance. The MATLAB results indicate a substantial increase in reducing turnaround and waiting times.	sorting cost is expensive

Table 1 Related work summary

CHAPTER THREE

Methodology

3.1 Overview

The focus of this section is on the strategies and techniques used to conduct the study to attain the research's goal. We are primarily concerned with determining what kind of cloud simulation toolkits are available and selecting the most suited ones. As a result, we'll talk about the CloudAnalyst simulator, which was used in this study.

3.2 Design and Modelling

Design and modeling of the overall architecture of the algorithm are proposed and developed in this section. The design phase is prepared by drawing a flow chart and writing a pseudo code of the proposed work. There are two choices to develop a model for the algorithm and test the performance of task scheduling algorithm, The first alternative for developing a prototype and testing the performance of a task scheduling algorithm is to do genuine tests in a real cloud environment; however, his necessitates the use of cloud infrastructure, which is expensive. The second alternative is to use a computer-based simulator.

There are numerous cloud simulators for testing the performance of task scheduling algorithms for cloud computing applications, some of them are *can cloud, DCSim* GridSim, *GreenCloud, MDCSim, GroudSi,* CloudAnalyst, and CloudSim. CloudAnalyst is a cloud simulator toolkit which is an extended version of the CloudSim tool. It supports us to simulate and estimate the performance of cloud services based on severe response time, data processing time, and resource utilization. In this simulation tool, response time and data center processing time act as performance evaluation parameters. Simulation results help us in the advancement of the quality of service (QoS) [38].

CloudAnalyst provides graphical user interface-based output that is simple to use and make modifications, its ability to save the output benefits that are

upgraded from Cloud-sim and other simulators. CloudAnalyst features simple user-friendly commands and displays for setup and use. It also comes with a Java programming language bundle installed. Editing and adding our task scheduling scripts is simple.[39]

This makes it easier for the researcher to integrate other existing comparison algorithms and the proposed algorithm. Some of the existing contrast algorithms are simply incorporated and preloaded into CloudAnalyst, whereas others must be written and integrated. Another advantage of the Cloud Analyst toolkit is the GUI (Graphical User Interface) based output, which includes tables (rows and columns), graphs, and charts, which are highly desired for reviewing a variety of results obtained during Cloud Analyst simulation [39]. These kinds of GUI-based presentations aid in understanding and identifying the important outlines of the parameters, as well as in comparing them. As a result, in the experiment section, its table and chart-based outputs facilitate the comparison and analysis mechanism.

The CloudAnalyst simulation includes the following metrics [40] [41]:

- ✓ The general response time, which includes the minimum, maximum, and average.
- ✓ General data processing time within the data center, including minimum, maximum, and average values.
- ✓ Response time for each user base, as well as minimum, maximum, and average values average.
- ✓ Each data center's minimum, maximum, and average time.
- ✓ The overall price of a virtual machine or nodes.
- ✓ The cost of each VM or data center.
- ✓ The cost of each data within a single data center.
- ✓ The total cost of each data center.

3.4 Research Methodology Development

Quantitative research frequently involves experiments, whereas qualitative research can implement methods such as field studies [8]. We have used both

quantitative and qualitative research methods to determine how the proposed algorithm is expected to support problem-solving solutions. We used qualitative research to deal with the description and understanding of the situation behind the task scheduling issues and data study, as well as quantitative research for performance metrics, numerical analysis of results, and demonstrating relationships between scheduling metrics for the study's evaluation.

3.5 Experimental Setup and Working Environments

In this part, we'll talk about working environments, which comprise both software and hardware needs. The next parts go into the software, hardware requirements, and programming language used for the development of the proposed program.

3.1 Software Requirements

The software that was used to conduct this study is listed below.

- Operating System: - Microsoft Windows 10
- IDE: - NetBeans IDE8.2
- Programming Language: - Java
- Simulation Tool: - CloudAnalyist

§ Operating System	- Microsoft Windows 10
§ IDE	- NetBeans IDE8.2
§ Programming Language	- Java
§ Simulation Tool	- CloudAnalyist

3.2 Hardware Requirements

The hardware requirement we used in this study is listed below.

- Processer: Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz 2.60GHz.
- RAM: Installed memory (RAM) 4.00GB.
- System Type: 64-bit operating system x64 -based processor.

3.3. Implementation

After we understand the CloudAnalyst simulator, the next step was to write a code for a proposed algorithm using java IDE, for which the java programming language was used because CloudAnalyst simulator architecture and implementation are developed in a Java-based environment [40]. In addition, the Java programming language has several features that are used in the development and deployment of a software environment for cloud computing.

Java, in particular, appears to be ideal for multi-model communication environments. Java's platform-independent byte code can run on a variety of platforms, making it a useful programming language for portable cloud computing. In addition, Java's ability to sequentially code has grown significantly in recent years. Java provides an advanced interactive graphical user interface framework as well as a model for invoking procedures on remote Fobjects [45]. So, to develop the proposed task scheduling algorithm, we first integrate the CloudAnalyst simulator with NetBeans Java IDE (integrated development environment) because it is very simple to import the available classes in the CloudAnalyst simulator, and then we develop the proposed algorithm.

3.4. Testing and Evaluation

After the configuration of the necessary information on the simulation tool, testing and evaluation is performed to determine how well the algorithm executes. To accomplish this, some evaluation criteria and techniques are considered, and the proposed algorithm is then compared to the base paper algorithm **(Round Robin Scheduling Algorithm Based on Analytic Model)** task scheduling algorithm by using different parameters. The parameters are Response time, data center processing time, number of context switches, and resource utilization. Finally, the experiment's findings are discussed, identified, and proposed as a solution.

CHAPTER FOUR

Proposed Work

4.1 Overview

As previously stated, we proposed an enhanced version of the round-robin scheduling technique in this paper. The selection of a time quantum is very hard and challenging because if it is too short, too many context switches will reduce CPU efficiency, on the other side if it is too long, heavy workloads will busy the processor, which results in a long average waiting time. Because of this, we concentrate on selecting the best TQ. For this, we propose a Task scheduling algorithm by considering variance among tasks weight. The main objective of this research is to create an effective task scheduling algorithm for cloud computing for the improvement of response time and better resource utilization. . This Chapter delivers a new Task scheduling algorithm by considering variance among tasks weight. It presents the VCRR (Variance Considered Round Robin) task scheduling algorithm design, implementation, and evaluation conditions. The design, implementation, and evaluation standards will be discussed in this chapter. In the design section, components of the proposed model with a brief description and a complete algorithm with pseudo-code are presented.

4.2 Proposed Algorithm

In this study, we proposed an improved algorithm Task scheduling algorithm that considers the variance among available processes in the queue. This algorithm first evaluates the mean and median of tasks, after that it finds the variance based on the burst time of tasks. Then it compares the mean and median of the tasks which are in the queue and chooses the greater one. To get the best Time Quantum (TQ) it compares the maximum of mean and median value with the variance. If the variance is greater than the maximum mean and median TQ will be evaluated by subtracting the root square of variance from

the maximum mean and median. If the variance is not greater then TQ will be evaluated by subtracting the root square of variance divided by several tasks from the maximum of mean and median.

Because the selection of a time quantum is very hard and challenging in the RR task scheduling algorithm a new technique was proposed in this study to cope with the blockage in the front of the round-robin algorithm. Modifying the round-robin algorithm to make dynamically control the time slice based on the conditions is a more feasible alternative for the standard RR and promising results regarding the response time. In this work, we will present the new proposed algorithm and analytical model round-robin algorithm and we will compare the results of two algorithms.

In the proposed approach two improvements have been made to the analytical model task scheduling algorithm.

1. The first improvement is done by removing data splitting into two parts (light weight, heavy weight). This modification will highly improve the performance of the system and resource utilization.
2. The second enhancement is done on finding the best Quantum (TQ). This enhancement is the core of this paper as it significantly improves the response time and waiting times of the tasks as well as the performance of the cloud server.

Therefore, we will have two algorithms which are the main algorithm (Algorithm 1) and TQ selecting algorithm (Algorithm 2). The main algorithm is designed for resource allocation in a cloud computing server. And the second algorithm is considered to choose the best Time quantum.

Algorithm 1: proposed Round Robin

1. Input an initial list of tasks.
2. Apply the analytic model which gives a sorted list in increasing order.
3. Find TQ using a new methodology (Explained in Algorithm2)
4. Set TQ =new TQ

5. Execute all tasks in the ready queue without dividing light and heavy tasks.

6. If a new task has arrived go to step 2

4.3 Pseudo code: for algorithm 1

```
1: Begin
2: Define Object function process (pi), p= (p1, p2, p3, p4…, pn) with their
   waiting and Burst Time
3: Initialize: process Pi (1, 2, 3…...n)
4: Check new available tasks in the ready queue and set NP=number of
   processes
5: Find TQ= Algorithm2 ()// output from algorithm 2
6: While [NP ≥ 0()]
7: for(counter=0;counter<NP; counter++) {
8: Execute processes in the queue with same TQ until new processes are
   arrived
9: If new processes are arrived
10:     Break; go to 4
11:     Else
12:     Execute all the tasks with the same TQ in the first cycle}
13:     Check p= (p1, p2, p3….pn)
14:     For each pi
15:     If (pi (BT ≤ TQ)
16:     Remove: Pi from the ready queue
17:     End if
18:     Else (pi (BT>Q)) and Pi remain in ready queue
19:     TQ = TQnew = Algorithm2()
20:     End For
21:     Repeat until all p=(p1,p2,p3….pn) are completed
22:     End while
```

Here, we present an explanation of the proposed algorithm. In our algorithm, the arrival time of process tasks has given a uniform standard. Whether all tasks have zero waiting time or different waiting times it does not matter in our algorithm. After defining and initializing processes the next step will be putting them in the ready queue including their waiting time and burst time.

Then the tasks in the queue are executed in the following way:

Steps:

- ✓ Check if new tasks have arrived in the ready queue. The list of processes in the ready queue will be sorted after applying the analytical model.
- ✓ Time Quantum (TQ) will be calculated using the new proposed algorithm (see algorithm 2)
- ✓ The list of processes will not be divided into a light and heavy list
- ✓ The CPU is allocated to processes using proposed RR scheduling with a
- ✓ Time Quantum (TQ) =new TQ (will be selected using algorithm2)
- ✓ The process is repeated until the list is finished or a new task has arrived.
- ✓ If a new task has arrived, the iteration of the loop will be stopped and go to step 1.
- ✓ If there is no new task all tasks in the queue will be executed and the queue will be empty

4.4 Proposed Algorithm for choosing best TQ

We have proposed variance considered Time Quantum finding method to get the best TQ for Round robin task scheduling algorithm. TQ selecting method Algorithm description is presented as follows. After finding the mean and median of burst time of all tasks in the ready queue the TQ will be calculated as follow.

$$\boldsymbol{If}\ \ \boldsymbol{var} > (\mathrm{Np} * \mathrm{Math.sqrt(mean)})$$

$$\boldsymbol{TQ} = \boldsymbol{Max}(\boldsymbol{Mean}, \boldsymbol{Median}) - \mathbf{3.14} * \sqrt{\boldsymbol{Var}}$$

$$\boldsymbol{Else}$$

$$\boldsymbol{TQ} = \boldsymbol{Max}(\boldsymbol{Mean}, \boldsymbol{Median}) + \mathbf{3.14} * \sqrt{\boldsymbol{Var}}\ /\boldsymbol{N}$$

Where: -

Var: is the variance of the available task in the queue,

Mean: is the mean value of all tasks in the list,

N: is the number of tasks in the queue.

Median: is the median value of all tasks in the list,

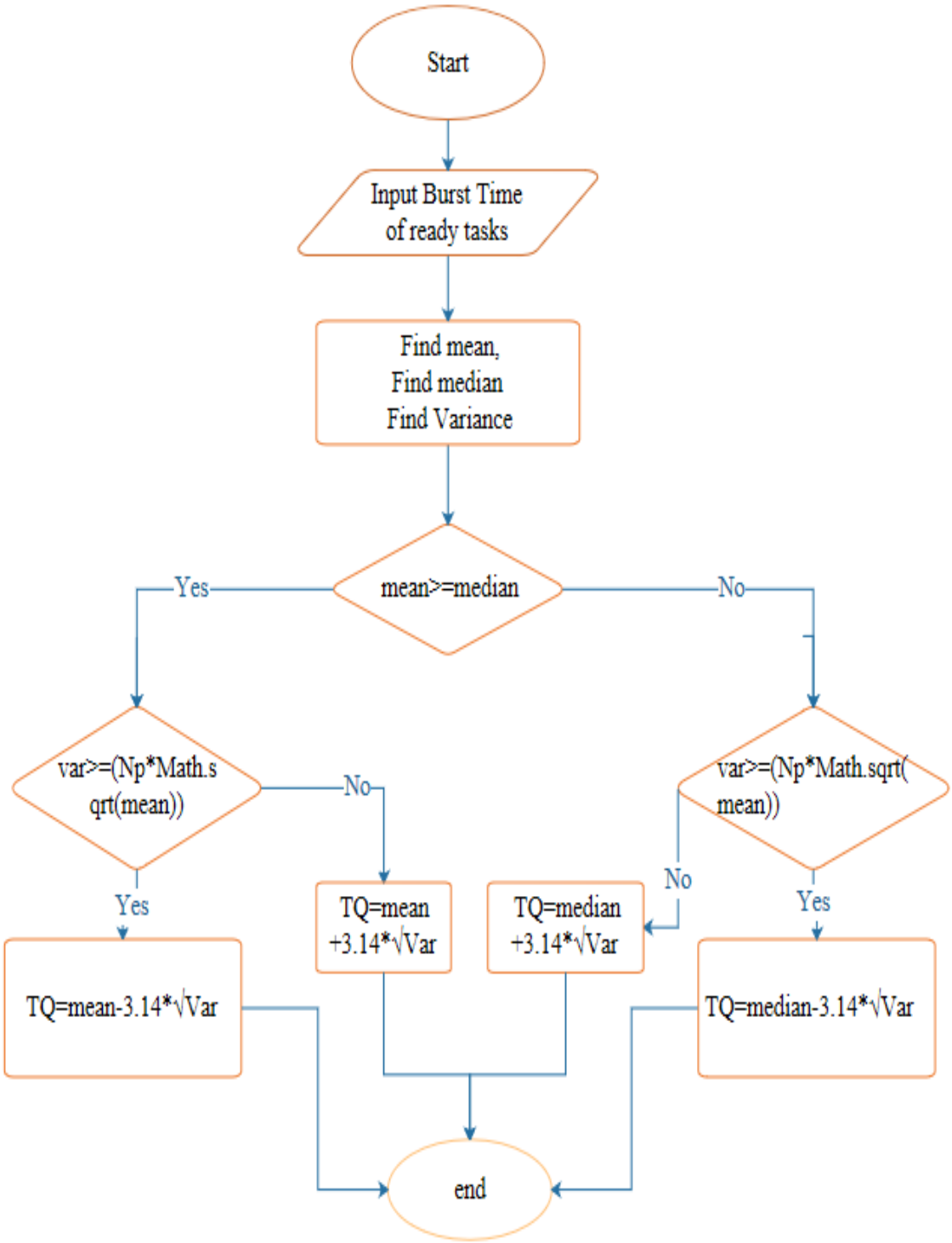
Algorithm 2: Proposed Algorithm for choosing best TQ

In the next lines, we explain a description of the proposed algorithm. Because our work is the improvement of existing work, we have used an analytical model in our algorithm.

1. Input initial values of burst time.
2. Find the median Burst time of the tasks. (Take the burst time of the middle task.)
3. Find the mean of the Burst time of the tasks.
4. Find median of all processes based on the Burst time of the tasks.
5. Find the variance (standard deviation) of Burst time
6. Compare mean and median
7. Choose the greater value from mean and median and assign it to temporary variable temp
8. If temp greater than the variance
9. Subtract from temp the square root of the variance and assign the result to QT.
10. Else Divide variance to the total number of arrived tasks and assign to temp2
11. Subtract temp2 from temp and assign the result to QT.

4.5. Pseudo code: for algorithm 2

1: Begin
2: Input Pi(BT)=burst time of each tasks
3: Np = number of process // number of ready tasks in the queue
4: If (NP=1)
5: TQ=BT/2
6: Else
7: mean=select the middle burst time of task in the ready queue
8: median= $Md = \left[\frac{\sum_{0}^{n-1} \quad Pi\,(BT)}{Np}\right]$ //the average burst time of all process in the queue
9: Var=standard deviation()
10: Max= $Max(Mean, Median)$
11: $If \;\; Var \geq Max(Mean, Median)$
12: $TQ = TQ\ new = \; Max(Mean, Median) - 3.14 * \sqrt{Var}$
13: $Else$
$TQ = \; TQ\ new = \; Max(Mean, Median) + 3.14 * \sqrt{Var}\ /N$



Flow chart 1: flow chart for selecting best TQ

CHAPTER FIVE

Results and Discussion

5.1 Overview

In this section, we present the results of the proposed approach on three different cases of processes. To show the success of our methodology and to assure a fair comparison between the proposed approach and existing algorithms, we demonstrate the same number of processes, same burst, and arrival time. We have selected three lists of processes to show the effectiveness of the proposed algorithm. The proposed approach works with any other list of examples. The list of processes we opted for this demonstration is low variance, medium variance, and high variance. The arrival time can be variable or zero arrival time. Ten processes P1, P2, P3, P4 P5, P6, P7, P8, and P10 have been considered.

5.2 Performance Analysis of proposed work

For the performance analysis, the two essential scheduling components of time and memory allocation are selected as control variables to develop an effective CPU scheduling based on the round-robin algorithm [42] The efficiency of the proposed algorithm is determined by the result of these control variables. However, it is necessary to mention that context switching is another important component of the round-robin algorithm that should be considered. As previously stated, context switching defines the CPU switching from one process or thread to another.

A process or task is a program that is executing or running in the CPU. The Round Robin method has difficulty with context switching since it causes a process to be swapped between the ready queue and the running state in the CPU numerous times before the task finished execution. If the process has a long burst time and only a small amount of quantum time is given, the problem becomes even worse. Therefore, reducing the number of context shifts is a very

important criterion. To reduce the number of contexts switch the round-robin method requires progressively changing the time quantum in a systematic method. The time quantum is the most significant component of the round-robin algorithm for the improvement of performance [42]

The structure of our algorithm is based on the dynamic allocation of the time slice. Ten processes were used to prove the working of the proposed algorithm, creating a ready queue with the ten processes labeled P1, P2, P3, P4, P5, P6, P7, P8, P9, and P10.

The lowness and highness of variance will determine using the following comparison.

If var >(Np*Math.sqrt(max(mean,median),

Where var: variance, Np number of tasks in the ready queue. we will consider the variance is high. If it is less then we will consider the variance is low.

5.3. Case 1: Medium Variance List of Tasks

In the first case we supposed these processes arrived 12, 5, 0, 55, 0, 0, 0, 0, 0 and 0 milliseconds (ms) respectively. The burst times (BT) of P1, P2, P3, P4, P5, P6, P7, P8, P9, and P10 were 328, 516, 617, 205, 24, 31, 624, 108, 152, and 998 ms, respectively. The processes were first arranged in ascending order using after applying the analytical model of using burst time and waiting time of processes.

Processes	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Burst Time	328	516	617	205	24	31	624	108	152	998
Arrival Time	12	5	0	55	0	0	0	0	0	0

Table 2: process burst time and Arrival time for case1

When we execute the all-selected algorithms for our experiment with the above-given data we got the following result in terms of waiting time. Even though there is no change in the seven processes, there is a big improvement in the rest of the three processes.

Waiting Time for	DPTQ	IRRAM	IRR	Proposed Algorithm	DPTQ vs PA	IRRAM vs PA	IRR vs PA
P1	508.00	303.00	508.00	508.00	0.00	-205.00	0.00
P2	843.00	638.00	843.00	843.00	0.00	-205.00	0.00
P3	1364.00	1159.00	1280.75	1205.85	158.15	-46.85	74.90
P4	260.00	2803.50	260.00	260.00	0.00	2543.50	0.00
P5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P6	24.00	24.00	24.00	24.00	0.00	0.00	0.00
P7	1981.00	1725.50	1713.50	1563.70	417.30	161.80	149.80
P8	55.00	55.00	55.00	55.00	0.00	0.00	0.00
P9	163.00	163.00	163.00	163.00	0.00	0.00	0.00
P10	2605.00	2292.00	2146.25	1921.55	683.45	370.45	224.70
AWT	780.30	916.30	699.35	654.41	125.89	261.89	44.94
Impt in %					16%	29%	6%

Table 3 Waiting time of each process for each algorithm case 1

As it is shown in table 3, there is a big improvement against all algorithms in terms of waiting time. Relative to DQTRR the proposed approach has improvement in task 3, task 7, and task 10, which have improvement is 158.15, 417.30and 683.45ms respectively with an average improvement of 125.89 which will be a 16% enhancement. In comparison with IVRR, the proposed approach has enhancement in task 4, task 7, and task 10, which have improvement is 2543.50, 161.80, and 370.45ms respectively with an average enhancement of 125.89 which will be a 29% enhancement. And in comparison, with IRR the proposed approach has also enhancement in task 3, task 7, and

task 10, have improvement is 2543.50, 161.80, and 224.70ms respectively with an average enhancement of 125.89 which will be a 16% enhancement.

The proposed approach has enhancement from DQTRR, IVRR, and IRR in 683.4ms, 370.45ms, and 224.70ms respectively. This improves the performance of the round-robin algorithm significantly as well as the execution of the proposed algorithm, as shown in Table 2. To make it clearer we add graph representation for this difference as shown below.

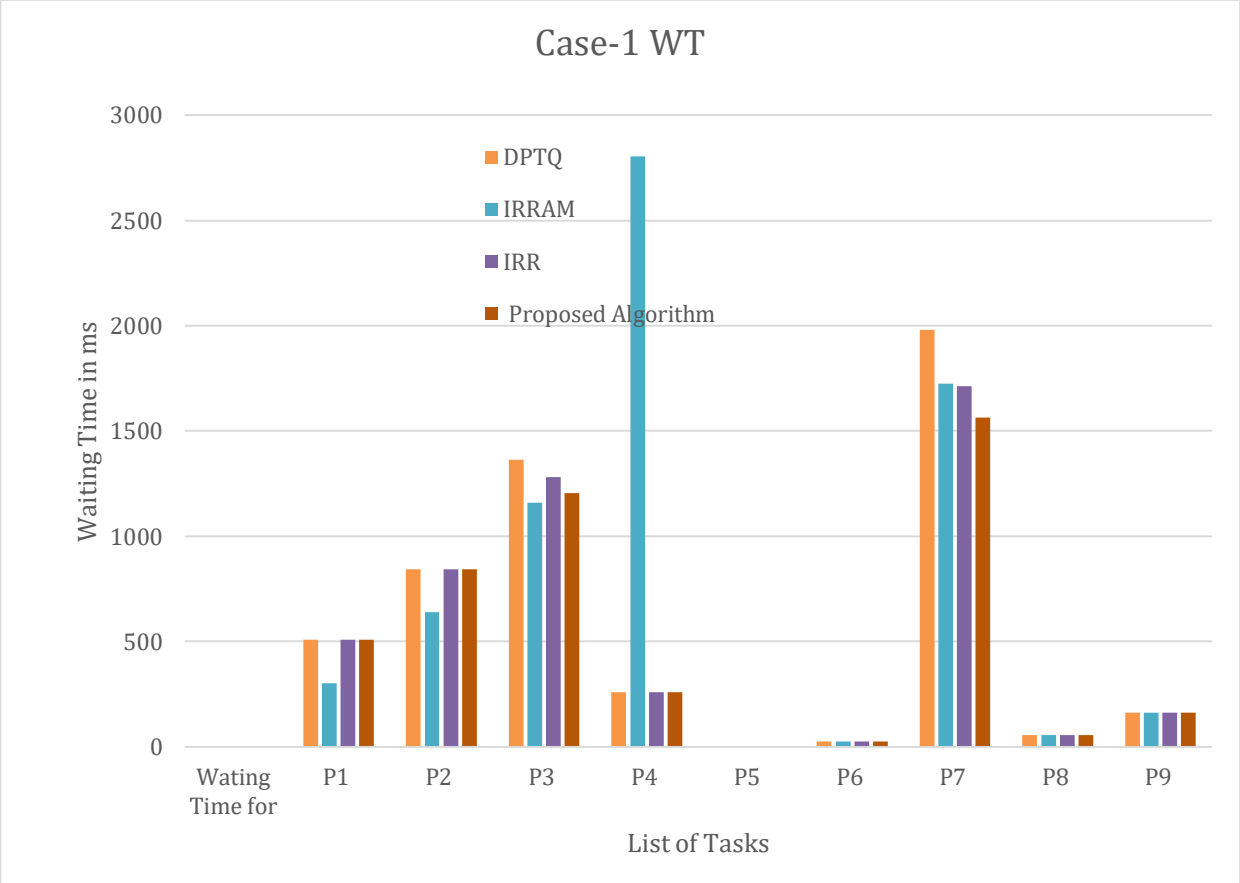
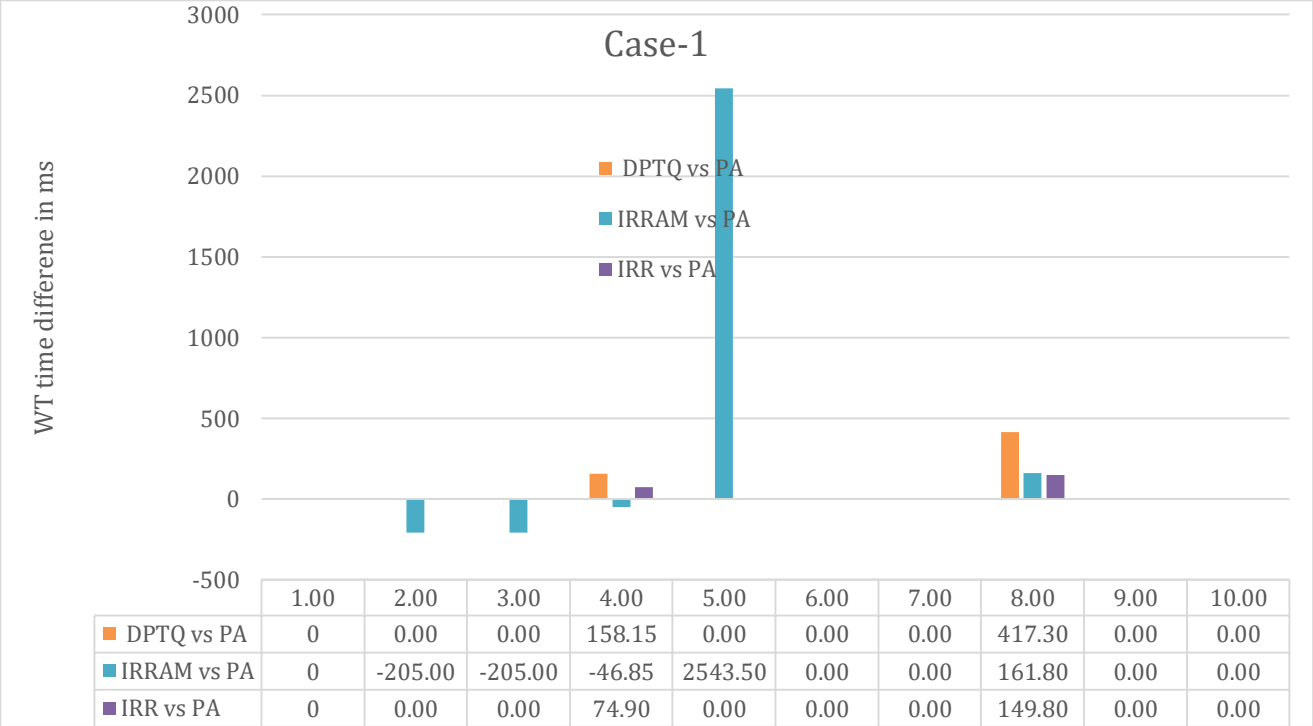


Figure 6: Graph representation of waiting time for each process in case 1

As shown in figure 6, the experiment result shows clearly in the graph that the proposed algorithm outperforms all algorithms in terms of waiting time.



	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00
DPTQ vs PA	0	0.00	0.00	158.15	0.00	0.00	0.00	417.30	0.00	0.00
IRRAM vs PA	0	-205.00	-205.00	-46.85	2543.50	0.00	0.00	161.80	0.00	0.00
IRR vs PA	0	0.00	0.00	74.90	0.00	0.00	0.00	149.80	0.00	0.00

Figure 7 Differences of waiting time in case 1

In figure 7, we present the improvement of waiting time (the difference of selected algorithms and proposed algorithm) for each process in milliseconds. The proposed approach has enhancement from DQTRR, IVRR, and IRR in 683.4ms, 370.45ms, and 224.70ms respectively. This improves the performance of the round-robin algorithm meaningfully as well as the execution of the proposed algorithm, as shown in Table 3. Next, we will present the improvement we made in terms of ATAT.

On the other hand, we get the following result in terms of Turnaround time (TAT) using the data listed in Table 1 for case 1. Here also there is a big improvement in the two algorithms.

TAT for	**DPTQ**	**IRRAM**	**IRR**	**Proposed Algorithm**	**DPTQ vs PA**	**IRRAM vs PA**	**IRR vs PA**
P1	836.00	631.00	836.00	836.00	0.00	631.00	205.00
P2	1359.00	1154.00	2657.25	2432.55	-1073.5	-1278.55	224.70

P3	1981.00	3114.00	2846.50	2696.70	-715.70	417.30	149.80
P4	465.00	3548.00	465.00	465.00	0.00	3083.00	0.00
P5	24.00	24.00	24.00	24.00	0.00	0.00	0.00
P6	55.00	55.00	55.00	55.00	0.00	0.00	0.00
P7	2605.00	3171.50	3037.75	2962.85	-357.85	208.65	74.90
P8	163.00	163.00	163.00	163.00	0.00	0.00	0.00
P9	315.00	315.00	315.00	315.00	0.00	0.00	0.00
P10	3603.00	3603.00	3603.00	3603.00	0.00	0.00	0.00
AWT	1140.60	1577.85	1400.25	1355.31	-214.71	306.14	65.44
Impt in %					-19%	19%	5%

Table 4: Turnaround Time (TAT) of each process in case 1

In terms of Turnaround Time also there is a great enhancement in our works. As it is shown in table 6, there is a great improvement in the two algorithms. But there is no enhancement regarding with DPTQ algorithm in terms of average turnaround time.

For IVRR and IRR there is an improvement, especially in IVRR in tasks 1, 3, 4, and task 7, the improvement is 631.00, 417.30, 3083.00, and 208.65 with the average improvement of 306.14 which is 19 percent of the base result. The proposed approach has improved by more than 306 ms in the first case. The third algorithm which is IRR also improved with an average of 65.44ms which is a 5 percent improvement. To show our improvement more clearly we present the turnaround time for each process and each task in the next graph.

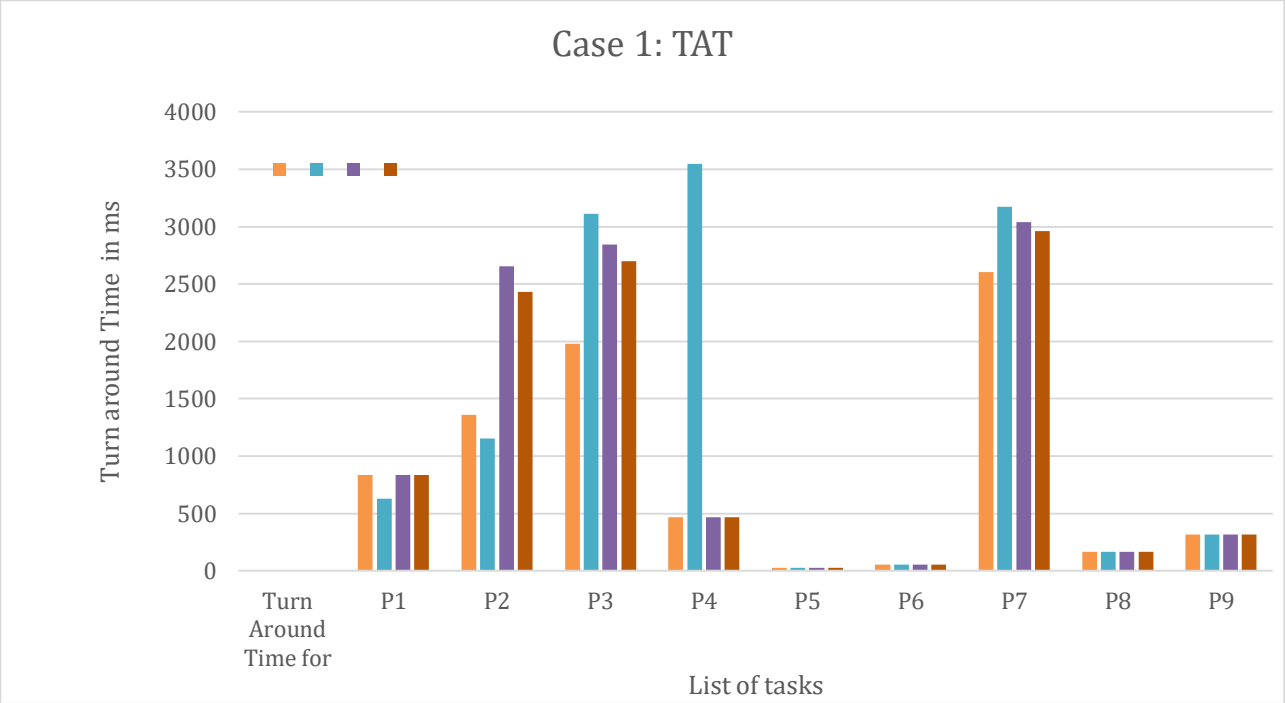


Figure 8 Turnaround time for each process in case 1

To review the enhancement, we achieved in our proposed work in case 1 which is a medium variance dataset we present the summary in table 5. As it is clearly shown in table 5 the proposed workout performs in all core scheduling metrics which are waiting time, turnaround time, and context switch.

	DPTQ	IRRAM	IRR	Proposed Algorithm
Average turnaround time (ms)	1140.60	1577.85	1400.25	1355.31
Average waiting time (ms)	780.30	916.30	699.35	654.41
No of context switch	11	16	15	15

Table 5: Average TAT and WT comparison table in case 1

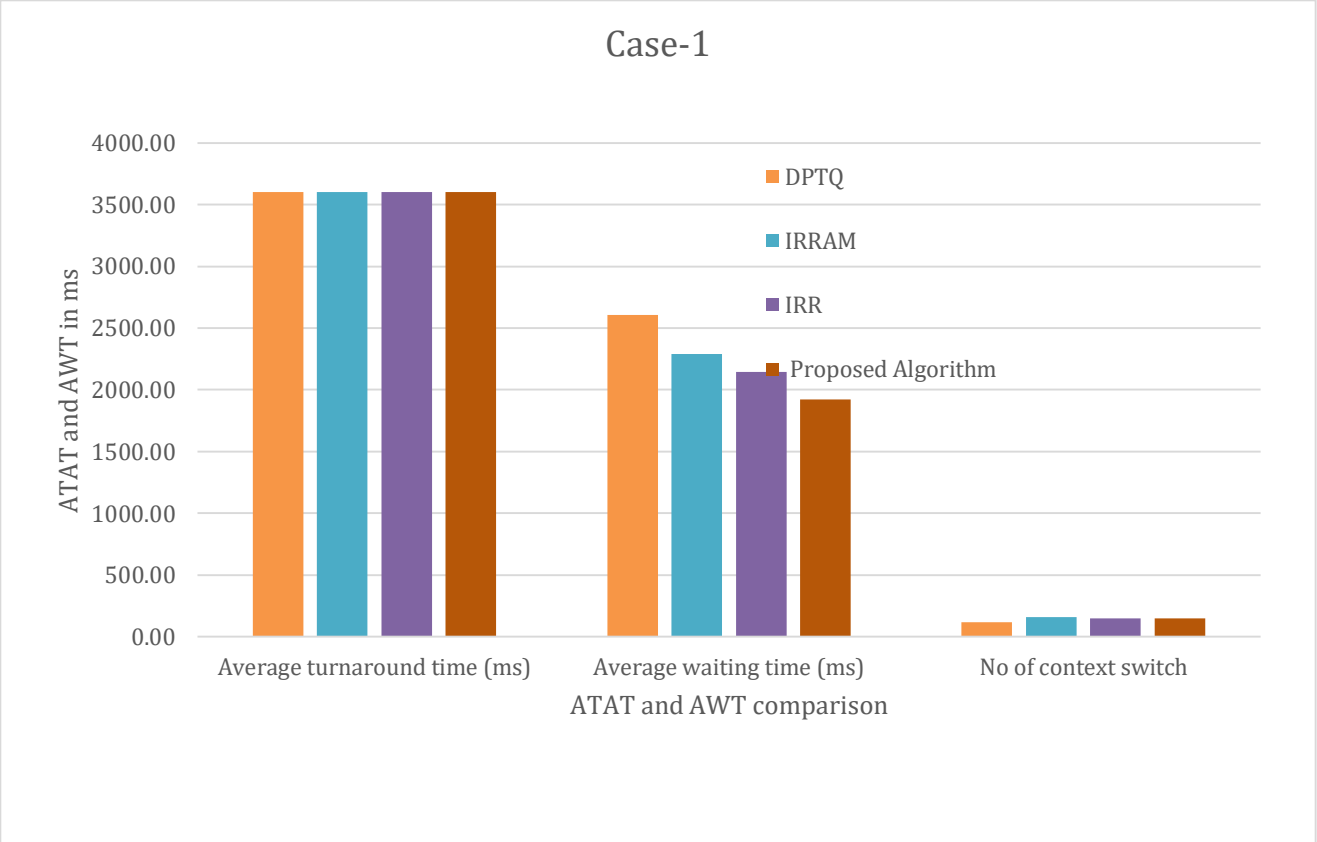


Figure 9 ATAT and WT for case 1

5.4. Case 2: Low Variance List of Tasks

In the second scenario we supposed these processes arrived 12, 45, 0, 25, 0, 0, 0, 0, 0,0 and 0 milliseconds (ms) respectively. The burst times (BT) of P1, P2, P3, P4, P5, P6, P7, P8, P9 and P10 were 17, 25, 24, 31, 24, 18, 12 and 28 ms, respectively.

Processes	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Burst Time	18	116	17	25	24	31	24	18	12	28
Arrival Time	12	45	0	25	0	0	0	0	0	0

Table 6: process burst time and waiting time for case 2

When we run the program using the above-given data, we got the following result in terms of waiting time for both algorithms. Here there is no difference in waiting time.

Waiting Time for	DPTQ	IRRAM	IRR	Proposed Algorithm	DPTQ vs PA	IRRAM vs PA	IRR vs PA
P1	35.00	142.00	35.00	35.00	0.00	107.00	0.00
P2	152.00	152.00	152.00	152.00	0.00	0.00	0.00
P3	12.00	12.00	12.00	12.00	0.00	0.00	0.00
P4	88.00	147.00	88.00	88.00	0.00	59.00	0.00
P5	89.00	47.00	89.00	89.00	0.00	-42.00	0.00
P6	166.00	123.00	166.00	166.00	0.00	-43.00	0.00
P7	65.00	71.00	65.00	65.00	0.00	6.00	0.00
P8	29.00	29.00	29.00	29.00	0.00	0.00	0.00
P9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P10	138.00	95.00	138.00	138.00	0.00	-43.00	0.00
Average	77.40	81.80	77.40	77.40	0.00	4.40	0.00
Impt in %					0%	5%	0%

Table 7: Waiting time of each process in base paper and proposed work for case 2

Table 7 results show that there is no enhancement in the two algorithms in terms of waiting time but there is a little bit of improvement regarding with IVRR algorithm.

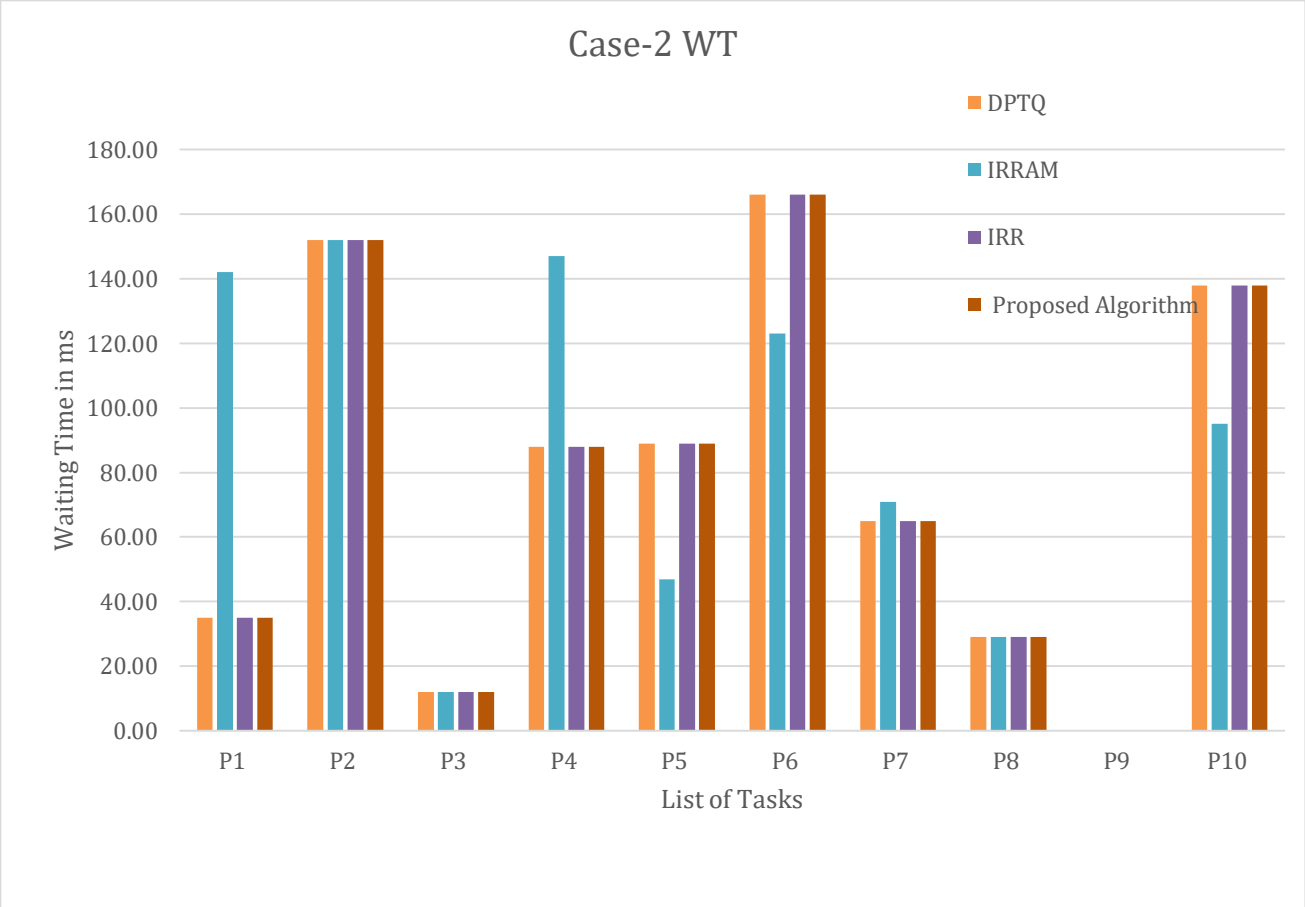
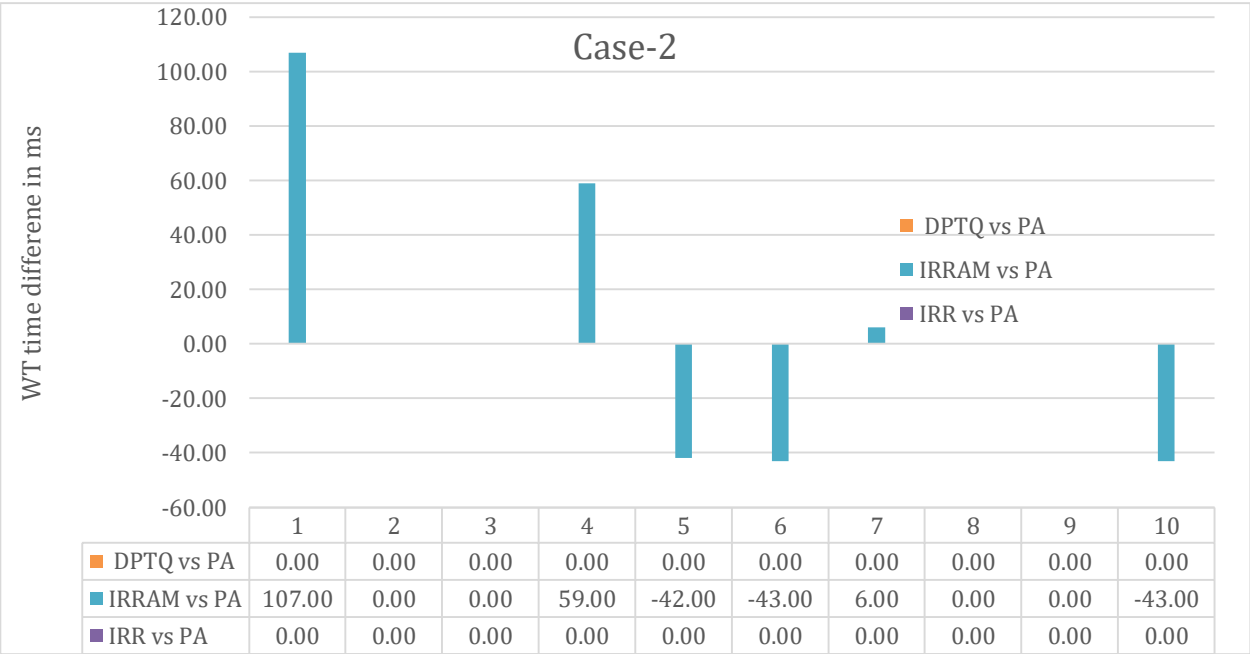


Figure 10 waiting time for each process and each algorithm in case 2



	1	2	3	4	5	6	7	8	9	10
DPTQ vs PA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IRRAM vs PA	107.00	0.00	0.00	59.00	-42.00	-43.00	6.00	0.00	0.00	-43.00
IRR vs PA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 11 WT time difference for case 2

As shown in Figures 10 and 11, there is no enhancement in all algorithms in terms of waiting time but there is a little bit of improvement regarding with IVRR algorithm.

Turn Around Time for	DPTQ	IRRAM	IRR	Proposed Algorithm	DPTQ vs PA	IRRAM vs PA	IRR vs PA
P1	53.00	160.00	53.00	53.00			
P2	268.00	523.00	268.00	268.00	0.00	255.00	0.00
P3	29.00	29.00	29.00	29.00	0.00	0.00	0.00
P4	113.00	172.00	113.00	113.00	0.00	59.00	0.00
P5	113.00	71.00	113.00	113.00	0.00	-42.00	0.00
P6	197.00	154.00	197.00	197.00	0.00	-43.00	0.00
P7	89.00	95.00	89.00	89.00	0.00	6.00	0.00
P8	47.00	47.00	47.00	47.00	0.00	0.00	0.00
P9	12.00	12.00	12.00	12.00	0.00	0.00	0.00
P10	166.00	123.00	166.00	166.00	0.00	-43.00	0.00
TAT	108.70	138.60	108.70	108.70	0.00	21.33	0.00
Impt in %					0%	15%	0%

Table 8: Turnaround Time (TAT) of each process in case 2

There is also no improvement in terms of Turnaround time (TAT) except for the second algorithm which is IVRR. The experiment result is presented in table 8. The improvement made from the IVRR algorithm is 15% in terms of average turnaround time,

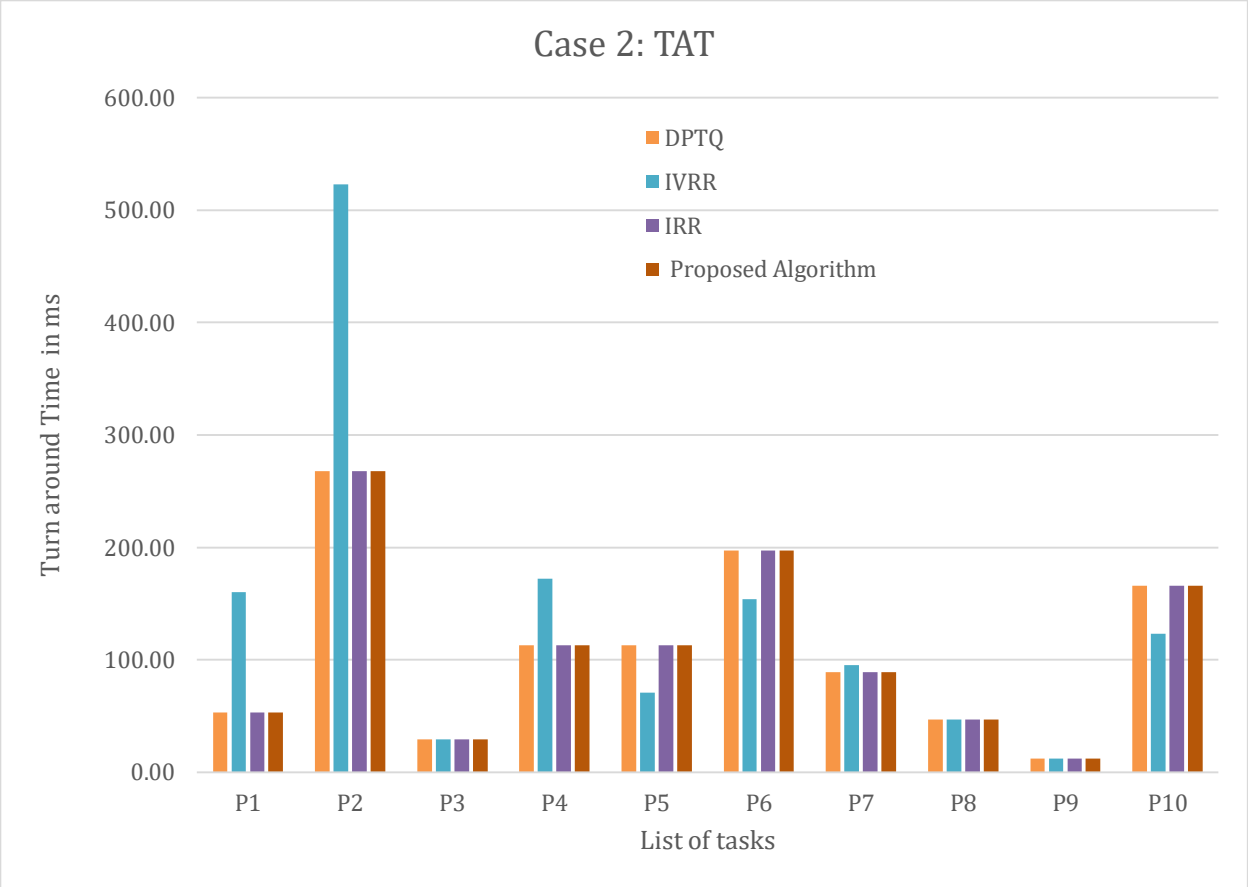


Figure 12 TAT for each process for case 2

	DPTQ	IRRAM	IRR	Proposed Algorithm
Average turnaround time (ms)	108.70	138.60	108.70	108.70
Average waiting time (ms)	77.40	81.80	77.40	77.40
No of context switch	11	16	13	12

Table 9: Average TAT and WT for base algorithm and proposed algorithm in case 2

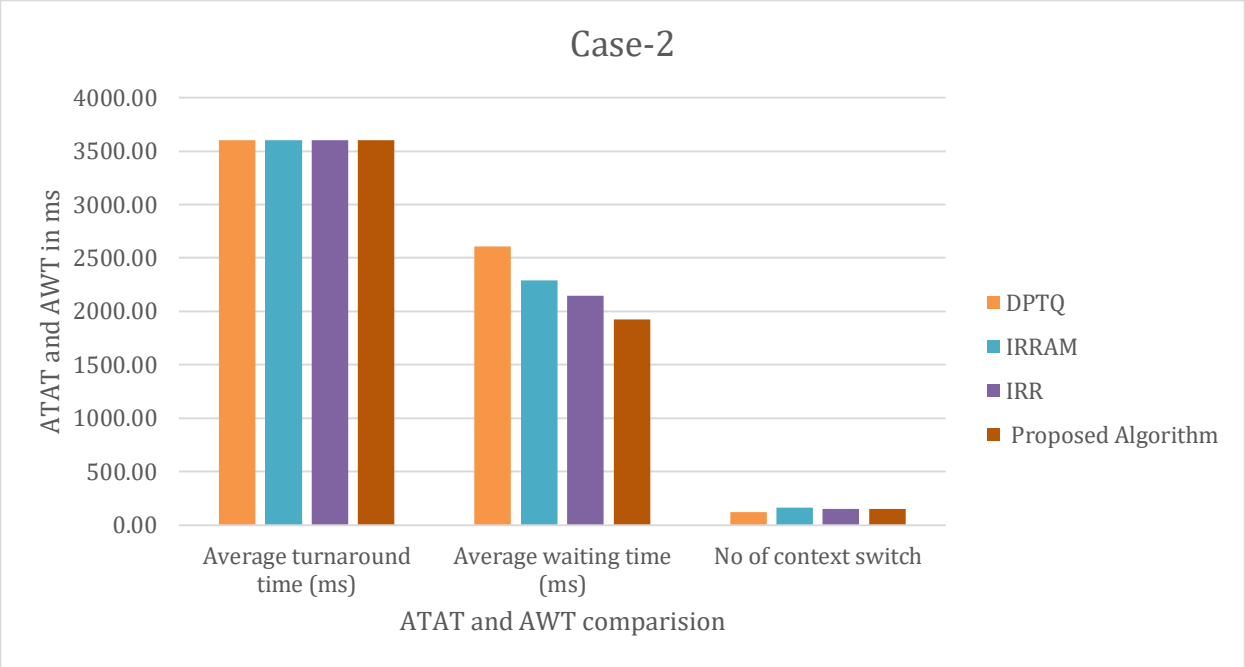


Figure 13 ATAT and WT for case 2

In the second scenario which is in the lowest variance data set the improvement is very small.

5.5. Case 3: High Variance List of Tasks

In the third case we supposed the processes have arrived 15, 0, 4, 0, 0, 0, 100, 0, 0 and 0 milliseconds (ms) respectively. The burst times (BT) of P1, P2, P3, P4, P5, P6, P7, P8, P9 and P10 are 70 , 300, 8000, 3365, 6000, 40, 701, 9890, 78 and 5653 milliseconds (ms) respectively.

Processes	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Burst Time	70	300	8000	3365	6000	40	701	9890	78	5653
Waiting Time	15	0	4	0	0	0	100	0	0	0

Table 10 list of processes for case 3

After an experimental demonstration with the above-given data, we got the following result in terms of waiting time. Here also, there is no change in some

of the processes, however, there is a remarkable improvement in the rest of the three processes.

WT for	DPTQ	IRRAM	IRR	P. Algorithm	DPTQ vs PA	IRRAM vs PA	IRR vs PA
P1	25.00	403.00	25.00	25.00	0.00	378.00	0.00
P2	188.00	118.00	188.00	188.00	0.00	-70.00	0.00
P3	16203.00	15328.50	13307.38	12805.75	3397.25	2522.75	501.63
P4	1189.00	488.00	1189.00	1189.00	0.00	-701.00	0.00
P5	10207.00	9506.00	8932.69	8681.87	1525.13	824.13	250.81
P6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P7	388.00	26885.50	388.00	388.00	0.00	26497.50	0.00
P8	23380.00	21159.00	17690.06	16937.62	6442.38	4221.38	752.44
P9	110.00	40.00	110.00	110.00	0.00	-70.00	0.00
P10	4554.00	3853.00	4554.00	4554.00	0.00	-701.00	0.00
Average	5624.40	7778.10	4638.41	4487.92	1136.48	3290.18	150.49
Impt in %					20%	42%	3%

Table 11 Waiting Time for each process for case 3

As it is shown in table 11, there is significant improvement in some processes the average waiting time improvement for DQTRR, IVRR, IRR is 1136.48, 3290.18, and 150.49 respectively. The proposed approach has an enhancement for DQTRR, IVRR, IRR is 1136.48, 3290.18, 150.49 respectively in this scenario. Execution of the proposed algorithm, as shown in Table 7. For more clarification, we showed up using graph representation for this difference as shown below.

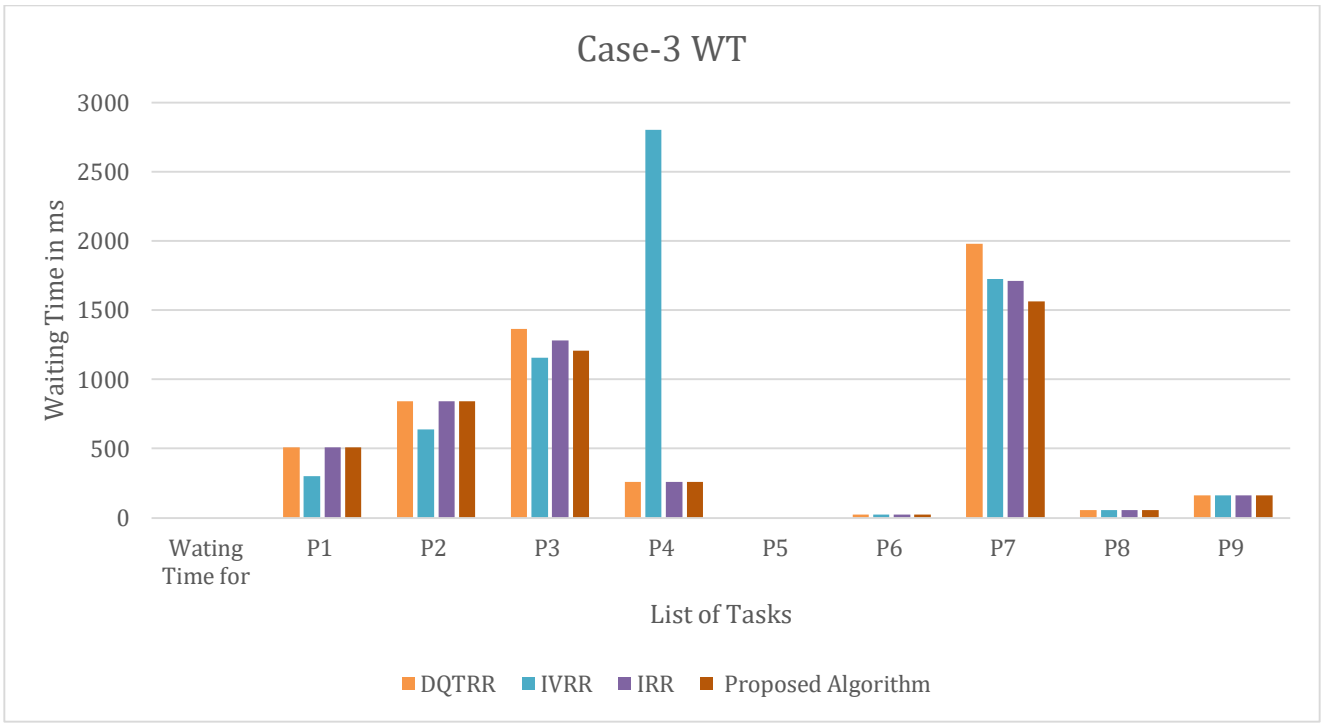
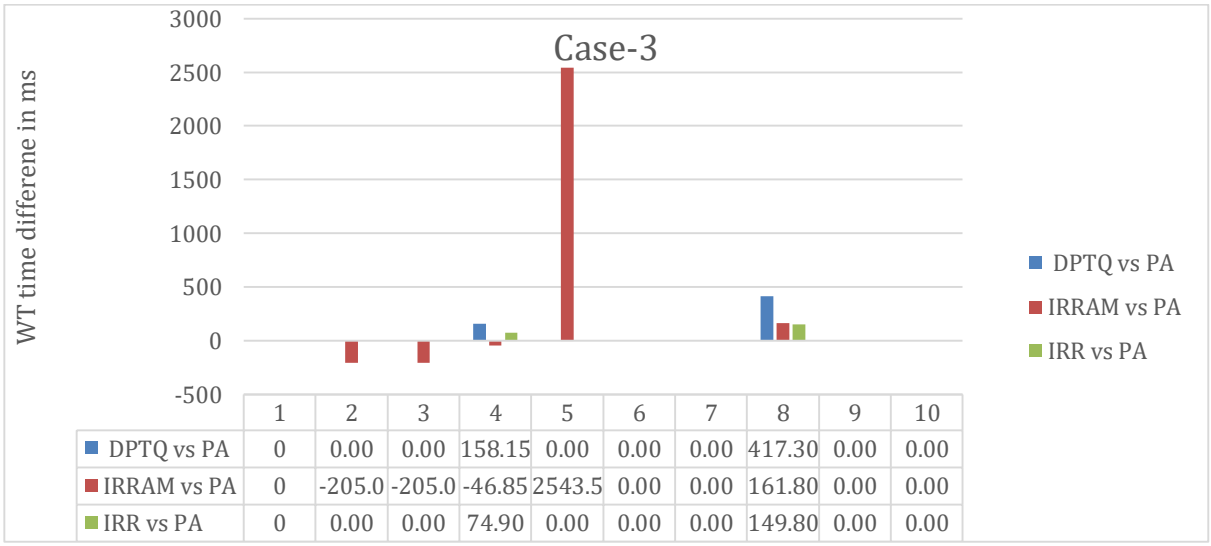


Figure 14 waiting time for each process case -3



	1	2	3	4	5	6	7	8	9	10
DPTQ vs PA	0	0.00	0.00	158.15	0.00	0.00	0.00	417.30	0.00	0.00
IRRAM vs PA	0	-205.0	-205.0	-46.85	2543.5	0.00	0.00	161.80	0.00	0.00
IRR vs PA	0	0.00	0.00	74.90	0.00	0.00	0.00	149.80	0.00	0.00

Figure 15 waiting time difference for each process in case 3

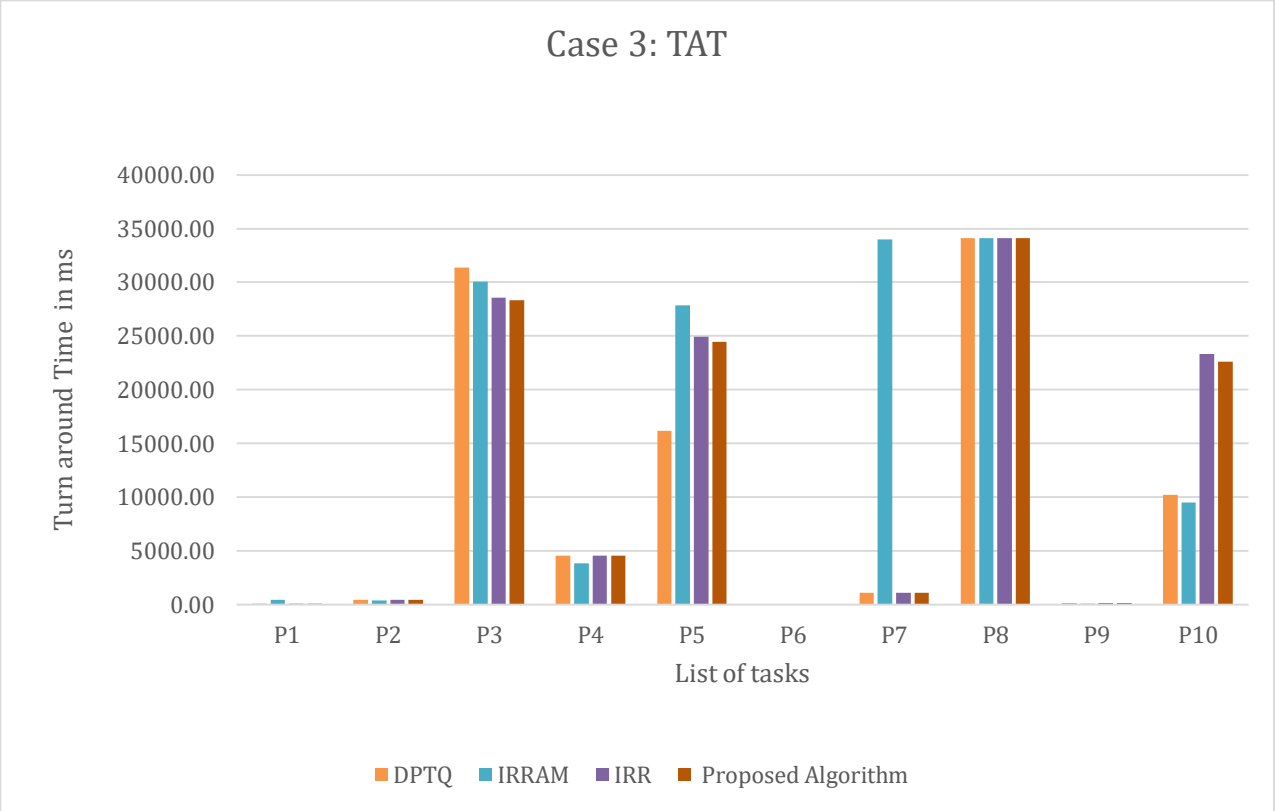


Figure 16 ATAT for case 3

On the other side, we get the following result in terms of Turnaround time (TAT) using the data listed in Table 7 for case 3. Here also there is a significant improvement in some processes.

Turn Around Time for	**DPTQ**	**IRRAM**	**IRR**	**Proposed Algorithm**	**DPTQ vs PA**	**IRRAM vs PA**	**IRR vs PA**
P1	95.00	473.00	95.00	95.00			
P2	488.00	418.00	488.00	488.00	0.00	-70.00	0.00
P3	31376.00	30029.50	28581.69	28330.87	3045.13	1698.63	250.81
P4	4554.00	3853.00	4554.00	4554.00	0.00	-701.00	0.00
P5	16207.00	27860.00	24964.38	24462.75	-8255.75	3397.25	501.63
P6	40.00	40.00	40.00	40.00	0.00	0.00	0.00
P7	1089.00	33997.00	1089.00	1089.00	0.00	32908.00	0.00
P8	34097.00	34097.00	34097.00	34097.00	0.00	0.00	0.00
P9	188.00	118.00	188.00	188.00	0.00	-70.00	0.00

P10	10207.00	9506.00	23343.06	22590.62	-12383.62	-13084.62	752.44
ATAT	9834.10	14039.15	11744.01	11593.52	-1954.92	2675.36	167.21
Impt in %					-20%	19%	1%

Table 12 ATAT for case 3

In terms of Turnaround Time also there is a great enhancement in our works. As it is shown in table 4, there is a great improvement in task 3, task 4, and task 7, the improvement is 79.88, 159.76, and 6650.15ms respectively. The proposed approach has improved by more than 6889.79 ms in the third case. This improves the performance of the round-robin algorithm significantly, as shown in Table 7.

	DPTQ	IRRAM	IRR	Proposed Algorithm
Average turnaround time (ms)	9834.10	14039.15	11744.01	11593.52
Average waiting time (ms)	5624.40	7778.10	4638.41	4487.92
No of context switch	12	16	150	15

Table 13 Result of Average WT and TAT for case 3

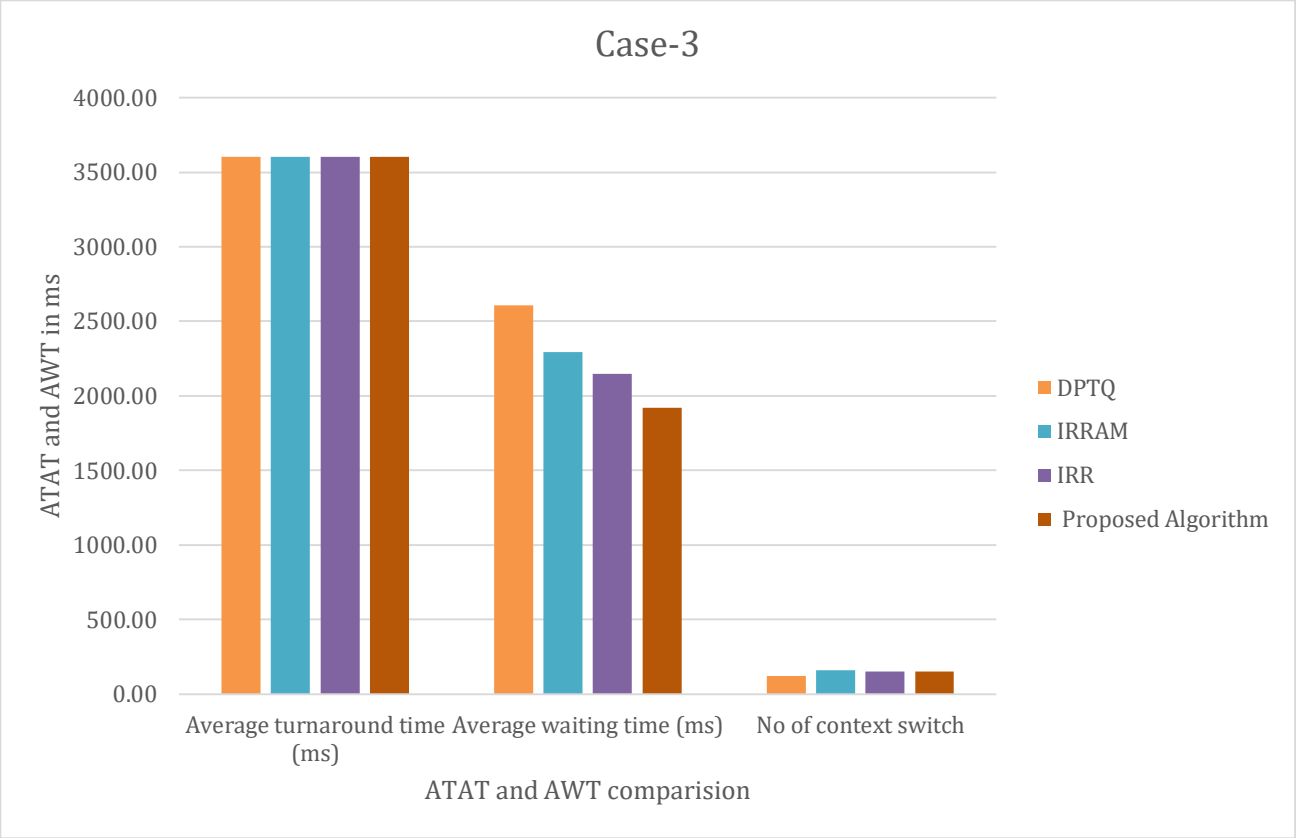


Figure 17 ATAT and AWT comparison in case 3

We demonstrated that our proposed approach outperformed all algorithms in all three scenarios. Especially in the first and third cases, our new proposed algorithm has brought significant improvement in waiting time, turnaround time, and context switch.

Conclusion

One of the most important issues that impact the overall performance of the cloud computing environment is task scheduling. The goal of a scheduling algorithm in a cloud computing environment is to schedule all types of tasks to reduce waiting time in the queue and the overall turnaround time spent to complete execution. This thesis's main contribution is to improve the round-robin algorithm by providing a new technique called VCRR. It focuses on providing a solution for the time quantum problem by calculating the time quantum based on the variance of burst time for all the tasks in the ready queue.

In this paper, we have worked to build a new algorithm that is better than the existing task scheduling algorithms in waiting time, turnaround time, and context switch, and we have accomplished it with very interesting and remarkable results. And we reco

Appendixes

Appendix A: source code for VCRR algorigthm.java

```
public void queue3_process(int ar_s1,int ar_s2)
   {

   main_counter2=main_counter;

      int d=0;
      int e=1;
     // cy=0;

        arr_size=ar_s2;

          for (d=0;d<ar_s2; d++){
        BT[d]=BT[d+ar_s1];//
        pr_index[d] =pr_index[d+ar_s1] ; // shift order of proccesses
        TAT[d]=TAT[d+ar_s1];}
      QT=value_TQ(ar_s2);
      QT_values[qt_count]=QT;
     qt_count++;

    // System.out.println("QT="+QT);
     counter=0;
  while(arr_size>=1){



     cy++;

    // System.out.println("cycle "+ cy+ " is safe");
     y=0;
    counter=0; // why?
     for(c=0;c<arr_size; c++){
```

```
        y=  pr_index[c];
                if (p_WT[y-1]==0) { // bemejemeriyaw cycle hulum silemidares,
keziya behuala ayasfeligm
                    y=  pr_index[c];
                    WT[y]=main_counter2;
                    p_WT[y-1]=1;
                }

       if(BT[c]>QT)// 20>10
         {
       BT[c]=BT[c]-QT;
      counter=counter+QT;
    main_counter2=main_counter2+QT;

       //  WT[pr_index[c]]= main_counter+counter; /// esti yifeteshal


         y=  pr_index[c];// yemayikeyerew index meyaza

     //    System.out.println("TAT of p"+y+"="+TAT_reserve[y]);
         TAT_reserve[y]=  TAT_reserve[y]+counter;  // kenesu  befit  yalewun
bicha new , mulu memzgeb alebet

      //   System.out.println("TAT of p"+y+"="+TAT_reserve[y]);


         }

         else
```

```
        {    counter=counter+BT[c]; // wating counter
         main_counter2=main_counter2+BT[c];
            BT[c]=0;// if Qt is greater than burst time

            // lik endecherese
              y=  pr_index[c];// yemayikeyerew index meyaza

            TAT_reserve[y]=TAT_reserve[y]+counter;

     }

         for(c=0;c<arr_size; c++)
    {
    if (BT[c]>0)
    {

             y=  pr_index[c];// yemayikeyerew index meyaza

    }

    }



  while(cc<arr_size) //

  {
       if( BT[cc]<=0)
     {
            y=  pr_index[cc];// yemayikeyerew index meyaza
       TAT_reserve[y]=TAT_reserve[y]+main_counter;//
```

```
    for(d=cc;d<arr_size-1; d++) // d=c mehon alebet, 0 kalihone beneberebt
endihon
    {

       BT[d]=BT[d+1];// shift all process one step to left

      pr_index[d] =pr_index[d+1] ; // shift order of proccesses



    }
   if (BT[d]>0) cc=cc+1;

  arr_size=arr_size-1;//
       }
     else cc=cc+1;



    public double value_TQ(int size)
  { int middle1, middle2,size1;
    double QT1=0;
    double temp1=0,temp2=0,avg=0,total=0;

     //  System.out.println("BT after:" +cy +"cycle");
       for (int i = 0; i <n ; i++)
     {          // System.out.println("P"+pr_index[i]+  ":BT=" +BT[i] );// ??
     }

    if (size==1 && cy==0) // yemejemeria cycle kehonena anid bicah kkehone
arrive yaderegew
    {
    QT1=BT[0]/2;
```

```
 return QT1; // return and break it
}
else
{
 if(size%2==0)
 {

    middle1=size1/2-1 ;// because index sijemr ke zero silehone
    middle2=size1/2;
   temp1= (BT[middle1]+BT[middle2])/2;
 }
 else
 {

    middle1=size1/2 ;
    temp1=BT[middle1];// median
   // System.out.println("median="+BT[middle1]);

 for (int i = 0; i <n ; i++)
  {
   total=total+BT[i]  ;
  }
avg=total/n;

temp2=avg; // mean

for (int i = 0; i <n ; i++)
  {
   sum=sum+ ((BT[i]-avg)*(BT[i]-avg))  ;
  }
double m=0;
if (n>1){
```

```
    m=sum/(n-1);

 stand_dev=Math.sqrt(m);}
else
{stand_dev=avg;}


 if (temp1>temp2)
 {
    QT1=temp1;
 }
 else {
    QT1=temp2;
 }

double df=0;
df=stand_dev/avg;


    if(stand_dev>=QT1)
         {
      QT1=QT1-Math.sqrt(stand_dev);
        // QT1=QT1+3.14*Math.sqrt(stand_dev);
         }
    else
    {
     QT1=QT1-Math.sqrt(stand_dev)/total_size;
     // QT1=QT1-Math.sqrt(stand_dev);
    }


        return QT1;
```

References

[1] Darabont Ö., Kiss K. J., Domokos J., “Performance Analysis of Remote Desktop Virtualization based on Hyper-V versus Remote Desktop Services”, in Proceedings of the 5 th International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (MACRo 2015), Tîrgu Mureș, Romania, 2015, pp. 125-135.

[2] A. T. Saraswathi, Y. R. A. Kalaashri, and S. Padmavathi, “Dynamic Resource Allocation Scheme in Cloud Computing,” Procedia Comput. Sci., vol. 47, pp. 30–36, Jan. 2015, doi:10.1016/j.procs.2015.03.180.

[3] Liu, J., Lai, W.: Security analysis of VLAN-based virtual desktop infrastructure. In: International Conference on Educational and Network Technology, pp. 301–304 (2010)

[4] Rodríguez-Haro F., Freitag F., L Navarro, Hernández-Sánchez E., Farías-Mendoza N., Guerrero-Ibanez J.A., González-Potes A., “A summary of virtualization techniques”, in Proceedings of the 2012 Iberoamerican Conference on Electronics Engineering and Computer Science, Procedia Technology 3, 2012, pp. 267 – 272.

[5] Taylor, M. E., Aboagye-Darko, D., & Shen, J. (2019). Cloud Management Systems - Load Balancing Algorithms and VDI Implementation Techniques. Artificial Intelligence and Security, 561–570. doi:10.1007/978-3-030-24265-7_48

[6] Arabinda Pradhan, Sukant Kishoro Bisoy and Pradeep Kumar Mallick, “Load Balancing in Cloud Computing: Survey” Springer 2020, https://doi.org/10.1007/978-981-15-2305-2_8

[7] Micheal Ernest Taylor and Jian Shen Cloud Management Systems and Virtual Desktop Infrastructure Load Balancing Algorithms - A Survey” 2017 https://doi.org/10.1007/978-3-319-68505-2_26

[8] Eduardo MagañaID, Iris Sesma, Daniel Morato´ ID, Mikel Izal "Remote access protocols for Desktop-as-aService solutions 2019

[9] Darabont Ö, Kiss K. J., Domokos J., “Cost effective computer infrastructure” in Proc. of the XXIVnd International Conference on Computers and Education (SZÁMOKT 2014), Székelyudvarhely, Romania, 2014, pp. 144-147.

[10] Dasilva, D.A., Liu, L., Bessis, N., Zhan, Y.: Enabling green it through building a virtual desktop infrastructure. In: Eighth International Conference on Semantics, Knowledge and Grids, pp. 32–38 (2012)

[10] Taneja, B. An empirical study of most fit, max-min and priority task scheduling algorithms in cloud computing ICCCA 2015, Greater Noida, India, 15–16 May 2015; pp. 664–667.

[11] Antonio Celesti, Davide Mulfari, Maria Fazio, Massimo Villari, Antonio Puliafito., “Improving Desktop as a Service in OpenStack”, in 2016 IEEE Symposium on Computers and Communication (ISCC)

[12] Washington Luna Encalada,* ID and José Luis Castillo Sequera, Model to Implement Virtual Computing Labs via Cloud Computing Services

[13] Survey and Analysis on Task Scheduling in Cloud Environment P. Akilandeswari* and H. Srimathi Indian Journal of Science and Technology, Vol 9(37), DOI:10.17485/ ijst/2016/v9i37/102058, October 2016

[14,] Shahbaz Afzal* and G. Kavitha “Load balancing in cloud computing – Ahierarchical taxonomical classification” Journal of Cloud Computing: Advances, Systems and Applications (2019)

[15] http://faculty.salina.k-state.edu/tim/ossg/Process/scheduler/metrics.html

[16] Mehwish Awan*, Muna Ali Shah A Survey on Task Scheduling Algorithms in Cloud Computing Environment, 2015

[17] Taghreed Balharith, Fahd Alhaidari “Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review” 2019 978-1-7281-0108-8/19/$31.00 ©2019 IEEE

[18,] Milani AS, Navimipour NJ (2016) Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends. J Netw Comput Appl 71:86–98

[19] http://www.e2matrix.com/blog/2018/01/12/types-of-scheduling-algorithms/ , (Accessed Aug 15, 2021)

[20] P. Pradhan, ―Modified Round Robin Algorithm for Resource Allocation in Cloud Computing, vol. 85, no. Cms, pp. 878–890, 2016

[21] F S. Elmougy, S. Sarhan, and M. Joundy, - A novel hybrid of shortest job first and round Robin with dynamic variable quantum time task scheduling technique, 2017.

[22] P. Sangwan, M. Sharma, and A. Kumar, -Improved Round RobinScheduling in Cloud Computing vol. 10, no. 4, pp. 639–644,2017

[23] Kaleeswaran, A., Ramasamy, V., & Vivekanandan, P. (2013). Dynamic scheduling of data using genetic algorithm in cloud computing. Park College of Engineering and Technology, Coimbatore, India.

[24] S. Banerjee, A. Chowdhury, S. Mukherjee, and U. Biswas, ―An Approach Towards Development of a New Cloudlet AllocationPolicy with Dynamic Time Quantum 1, vol. 52, no. 3, pp. 208–219, 2018

[25] B. Fataniya and M. Patel, ―Dynamic Time Quantum Approach to Improve Round Robin Scheduling Algorithm in Cloud Environment, vol. 4, no. 4, pp. 963–969, 2018.

[26] S. Banerjee, M. Adhikari, and U. Biswas, ―Design and analysis of an efficient QoS improvement policy in cloud computing, Serv.Oriented Comput. Appl., vol. 11, no. 1, pp. 65–73, 2017

[27] Chronopoulos, A.T.: Game Theory Based Load Balanced Job Allocation. http://graal. enslyon.fr/*lmarchal/aussois/slides/chronopoulos.pdf. Accessed 04 Dec 2018

[28] Lunsford, D.L.: Virtualization technologies in information systems education. J. Inf. Syst.Educ. 20(3), 339–348 (2017)

[29] Durgesh Patel, Mr. Anand S Rajawat “Efficient Throttled Load Balancing Algorithm in Cloud Environment” (IJMTER) Volume 02, Issue 03, [March - 2015]

[30] Balharith, T.; Alhaidari, F. Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review. In Proceedings of the 2nd International Conference on Computer Applications and Information Security, ICCAIS 2019, Riyadh, Saudi Arabia, 1–3 May 2019; pp. 1–7.

[31] M. D. Shah, A. A. Kariyani and D. L. Agrawal, "Allocation of Virtual Machines In Cloud Computing Using Load Balancing Algorithm," International Journal of Computer Science and Information Technology & Security, vol. III, no. 1, 2013.

[32] Shahera Saad Ali, Asmaa Jameel Al Nawaiseh, Yehia Helmy, Ibrahim Fathy Moawad and Emadeldin Khalil, A Proposed Model Based on Cloud Computing Technology to Improve Higher Education Institutions Performance, International Journal of Advanced Research in Engineering and Technology, 11(9), 2020, pp. 612-628.

[33] Jbara, Y.H. A new improved round robin-based scheduling algorithm-a comparative analysis. In Proceedings of the 2019 International Conference on Computer and Information Sciences, ICCIS 2019, Sakaka, Saudi Arabia, 3–4 April 2019; pp. 1–6.

[34] Analysis of CPU Scheduling Algorithms for Cloud Computing Ramasubbareddy Somula, Sravani Nalluri, M. K. NallaKaruppan, S. Ashok and G. Kannayaram, 2019

[35] Sudhansu Bala Das1 Sugyan Kumar Mishra2 Anup Kumar Sahu“ An efficient average execution time-round-robin (AET-RR) scheduling algorithm” Bharati Vidyapeeth’s Institute of Computer Applications and Management 2019

[36] Matarneh RJ (2009) Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes. Am J Appl Sci 6(10):1831–1837

[37] Tahani Aladwani (April 23rd 2020). Types of Task Scheduling Algorithms in Cloud Computing Environment, Scheduling Problems - New Applications and Trends, Rodrigo da Rosa Righi, IntechOpen, DOI: 10.5772/intechopen.86873 /71902

[38] B. Wickremasinghe, R. N. Calheiros and R. Buyya1, Cloud Analyst: A CloudSim based Visual Modeler for Analyzing Cloud Computing Environments and Applications, The University of Melbourne, Australia.

[39] Malhotra, Rahul and P. Jain, "Study and Comparison of CloudSim Simulators in the Cloud Computing," The SIJ Transactions on Computer Science Engineering and its Applications (CSEA), 2013

[40] D. Kashyap and J. Viradiya, "Service Broker Algorithm for Cloud-Analyst," International Journal of Computer Science and Information Technologies, vol. III, no. 2, 2014

[41] R. K. Mishra and S. N. Bhukya, "Cloud Computing-Software as Service," International Journal of Cloud Computing and Services Science (IJ-CLOSER), vol. 1, no. 1, 2012.

[42] Mohammad M. Tajwar, CPU Scheduling with a Round Robin Algorithm Based on Effective Time Slice, August 2017 https://doi.org/10.3745/JIPS.01.0018

[43] Aditya Makwe, Priyesh Kanungo A Survey of Scheduling Policies in Cloud Computing Environment, International Journal of Computer Trends and Technology (IJCTT) – Volume 34 Number 4 - April 2016, ISSN: 2231-2803 http://www.ijcttjournal.org Page 169

[44] Huajin, S., Deyuan, G., Shengbing, Z., Danghui, W.: Design Fast Round Robin Scheduler inFPGA. 0-7803-7547- 5/021/$17.00 @ 2002 IEEE

[45] Fahd Alhaidari 1, and Taghreed Zayed Balharith 2 Enhanced Round-Robin Algorithm in the Cloud ComputingEnvironment for Optimal Task Scheduling, 2021

[46] Bhavin Fataniya, Manoj Patel Dynamic Time Quantum Approach to Improve Round Robin Scheduling Algorithm in Cloud Environment 2018