GREEN CLOUD-ENERGY MANAGEMENT THROUGH EFFICIENT RESOURCE ALLOCATION SCHEME

Final report of the project (dissertation) is being submitted for partial fulfillment of the requirements for the degree of

Bachelor of Technology (B. Tech.)

in

***Information Technology** (IT)*

from Maulana Abul Kalam Azad University of Technology, West Bengal

by

ARPON BANDYOPADHYAY
Student of B. Tech. (IT) – Eighth Semester
Academic Session: 2022 - 2023
Registration No: 036244 of 2019-20
University Roll No: 11700219053

PRITI SARKAR
Student of B. Tech. (IT) – Eighth Semester
Academic Session: 2022 - 2023

Registration No: 035960 of 2019-20
University Roll No: 11700219082

SHIMLI CHAKRABORTY
Student of B. Tech. (IT) – Eighth Semester
Academic Session: 2022 - 2023
Registration No: 201170100220009 of 2020-21
University Roll No: 11700220106

SHREYA ROY

Student of B. Tech. (IT) – Eighth Semester
Academic Session: 2022 - 2023
Registration No: 201170100220001 of 2020-21
University Roll No: 11700220114

under the supervision of

Indrajit Pan, Ph. D. (Engineering)
Professor
Department of Information Technology
RCC Institute of Information Technology, Kolkata
MAKAUT, WB - College Code: 117

at

RCC INSTITUTE OF INFORMATION TECHNOLOGY
Canal South Road, Beliaghata, Kolkata, 700015
Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal (Erstwhile WBUT)
Accredited by AICTE, New Delhi, India
May – 2023

RCC INSTITUTE OF INFORMATION TECHNOLOGY
Canal South Road, Beliaghata, Kolkata, 700015

Affiliated to Maulana Abul Kalam Azad University of Technology,
West Bengal(Erstwhile WBUT)
Accredited by AICTE, New Delhi, India

CERTIFICATE

The dissertation titled Green cloud - energy management through efficient resource allocation scheme submitted by **Priti Sarkar** (University Roll No.:11700219082), **Arpon Bandyopadhyay** (University Roll No.:11700219053), **Shreya Roy** (University Roll No.: 1700220114), **Shimli Chakraborty** (University Roll No.: 11700220106) of B. Tech. (IT) 8th Semester (Academic session: 2022 – 2023) has been prepared for the partial fulfillment of the requirements for B. Tech. (IT) degree from Maulana Abul Kalam Azad University of Technology, West Bengal (formerly known as WBUT) under my guidance. The work hasn't been produced before for any degree or diploma elsewhere.

The report is hereby forwarded.

Date: *09th May, 2023*

OPTIONAL IN CASE

[Name of Guide]
Designation
Institute Name
(External Supervisor)
Dept. of IT
Countersigned by

Indrajit Pan, Ph. D.
(Engg.)Professor &
Head Dept. of IT RCCIIT,
Kolkata (Supervisor)

……………………………………………..
Name of Head
Department Information Technology
RCC Institute of Information Technology, Kolkata – 700 015, India

ACKNOWLEDGEMENT

We express our sincere gratitude to my supervisor Dr. Indrajit Pan, Professor and Head, Department of Information Technology, RCCIIT for extending his tremendous support and guiding us to take up this problem as our final year dissertation. We have been immensely benefited by his continuous follow up, rigorous monitoring and technical support.

We are also indebted to Dr. Anirban Mukherjee, Principal (Officiating), RCCIIT and Dr. Abhijit Das, Associate Professor, RCCIIT for guiding us at our initial stage of project and arranging necessary infrastructural support from the Department.

It is needless to mention yet we want to express our sincere gratitude to our other faculty members of the Department and our program coordinator, Dr. Soumen Mukherjee for helping us all along the tenure of our study.

We are thankful to Mr. Jayanta Datta and Mr. Amit Khan, respected faculty members from our department for their inspiring words and support all along the study tenure.

We must thank our classmates for their continuous support and all of them were very kind and helpful, hence it is difficult to pick any particular name for special acknowledgement.

Date: *09th May, 2023*

(Priti Sarkar)
Reg. No.:035960 of 2019-20 Roll No.: 11700219082

(Arpon Bandyopadhyay)
Reg. No.:036244 of 2019-20, Roll No.: 11700219053

(Shreya Roy)
Reg. No.:201170100220001 of 2020-21 Roll No.: 11700220114

(Shimli Chakraborty)
Reg. No.:201170100220009 of 2020-21 Roll No.: 11700220106

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Canal South Road, Beliaghata, Kolkata, 700015

Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal(Erstwhile WBUT)
Accredited by AICTE, New Delhi, India

CERTIFICATE of ACCEPTANCE

The dissertation titled Green cloud - energy management through efficient resource allocation scheme submitted by **Priti Sarkar** (University Roll No.:11700219082), **Arpon Bandyopadhyay** (University Roll No.:11700219053), **Shreya Roy** (University Roll No.: 1700220114), **Shimli Chakraborty** (University Roll No.: 11700220106) of B. Tech. (IT) 8th Semester (Academic session: 2022 – 2023) is hereby recommended to be accepted for the partial fulfillment of the requirements for B. Tech. (IT) degree from Maulana Abul Kalam Azad University of Technology, West Bengal (formerly known as WBUT)

Name of the Examiner	Signature with Date
1.	....
2.	....
3.	....
4.	....

TABLE OF CONTENTS

1. Introduction

Cloud computing is the delivery of different services like servers, data storage (cloud storage), databases, networking, and development tools over the internet which are located in a remote database. Resource Management (RM) in cloud computing refers to techniques that focus on the efficient sharing of cloud resources among multiple users. A cloud computing architecture is shown in fig 1.1.

Green cloud refers to using cloud services in a way to reduce its impacts on the environment. It focuses on designing clouds with green characteristics like load balancing, energy optimization, algorithmic efficiency, virtualization, reusability, recyclability etc.

This project aims to reduce the negative impacts of cloud computing on the environment due to the high amount of energy consumption by the cloud data centers. It focuses on using the least number of servers and providing the maximum output. This project uses Round Robin Algorithm to achieve this by scheduling the servers so that different servers are allocated at different arrival time.

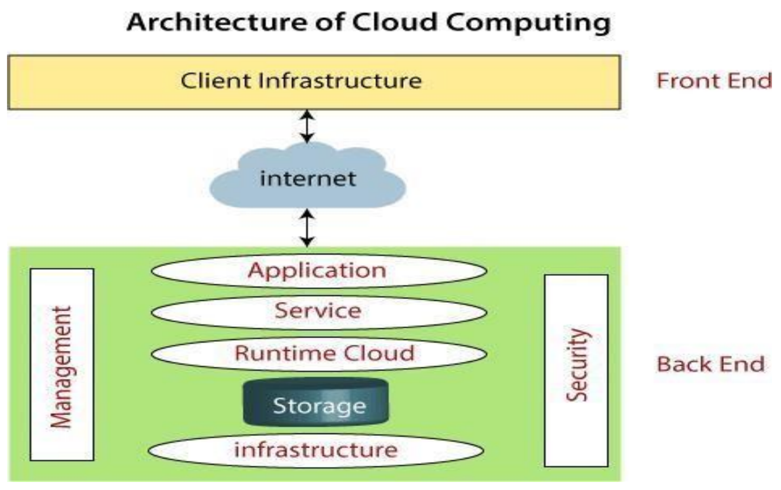


Fig No: 1.1 Cloud Computing Architecture [11]

1.1 Types of cloud models

Public Cloud: As shown in fig.1.2,Public clouds are the most prevalent and familiar cloud computing model and all the resources needed to run the infrastructure of servers, storage, networking components and supporting software are managed by the third party provider and accessed by users over the Internet via a web browser. Examples of public clouds are Amazon Web Services, Microsoft Azure and Google Cloud Platform.

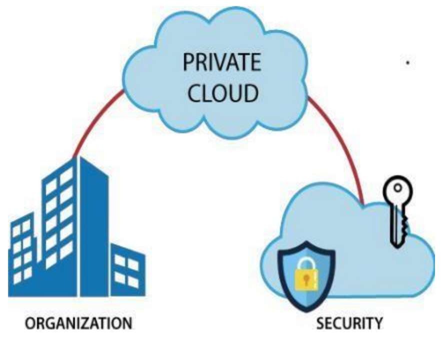
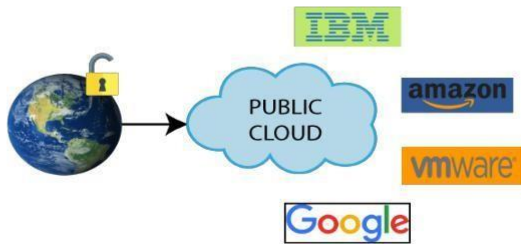


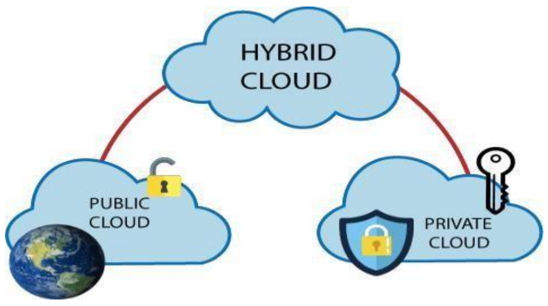
Fig No.1.2 Private Cloud [12]

Private Cloud Private Cloud is a computing framework fatal to use by a single organization, shown in fig.1.3. It can be regained at your own data center facility or at that of a third party supplier. The defining characteristics is that the resources are run and maintaining on a private network protocols alone. Unlike public cloud, private, the cloud is not shared with anyone else. This makes customizations and the regulatory compliance easier to manage, which is why private clouds are often utilized by
Financial institutions, government agencies and heavily regulated industries that need to exercise a high degree of control over their workloads.



FigNo.1.3 Public Cloud [12]

Hybrid Cloud: Hybrid cloud is a cloud computing environment that uses a mix of promises, private cloud and third party public cloud services with proportion between the two platforms, hybrid cloud combines public and private cloud resources to use the advantages of both. By allowing more clouds to move between private and public cloud as computing needs and cost change, hybrid cloud gives business greater flexibility and more data deployment options. A model of hybrid cloud is shown in fig.1.4.



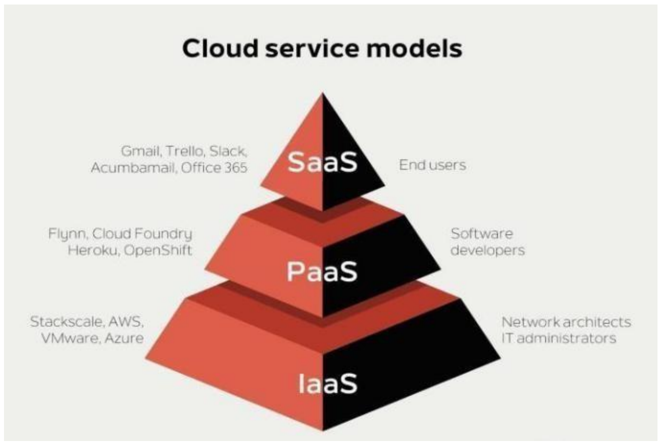
FigNo.1.4 Hybrid Cloud [12]

1.2 Different Cloud Service Models

- **IaaS (Infrastructure as a Service):** IaaS is also known as **Hardware as a Service (HaaS).**This is an instant computing infrastructure, provisioned and managed over the Internet. It is one of the four types of cloud services, along with software as a service, platform as a service and compromise as quickly scales up and down with demand, letting us pay only for what we use.
- **PaaS (Platform as a Service):** Platform as a service bus is a complete development and deployment environment in the cloud with resources that enable us to deliver everything from simple cloud based apps to sophisticated, cloud enabled enterprise applications, appreciate the resources we need from a cloud service provider on a pay as we go and access them over.

- **SaaS (Software as a Service):** Software as a service is one of the models of cloud computing services, which allows users to connect to cloud based apps and use them over the Internet.

A figure showing all the cloud service models is shown in figure 1.5.


Cloud service models
Gmail, Trello, Slack, Acumbamail, Office 365
SaaS
End users
Flynn, Cloud Foundry Heroku, OpenShift
PaaS
Software developers
Stackscale, AWS, VMware, Azure
IaaS
Network architects IT administrators

FigNo.1.5 Cloud Services Model (SaaS , PaaS , IaaS)[13]

2. Review of Literature

Different methods needed to improve energy efficiency for assisting cloud operators, decision-makers, and researchers in developing appropriate energy evaluation strategies are discussed in article [1], because energy consumption is increasing along with the increase in the digital economy. There is increasing attention on data centers for the accomplishment of energy consumption and energy reduction. Effective energy evaluation methods are required for the data center industry to achieve its objectives of reducing the demand for electricity.

Discussion about the improvement of some of the traditional algorithms which are energy awarded and have high performance is mentioned in the article [2].This is done without greatly reducing the performance, and improving the energy efficiency at the same time. A new algorithm named Energy-Performance Trade-off Multi-Resource Cloud Task Scheduling Algorithm (ETMC TSA) is designed and users can flexibly manage and control the energy and performance of a cloud system by tuning the probability parameter of the algorithm.

The various methods for energy consumption and improvement of performance mentioned in article [3], because of the large amount of energy consumed by data centers and consequently affecting our environment. Energy efficiency evaluation methods are done at three levels:-hardware, resource management, and applications. The economic concern is to reduce the operational costs of powering and cooling large-scale data centers with the large-scale uptake of cloud data centers to host industrial applications. Power management techniques are the methods that are used to optimize the energy efficiency of computer techniques.

Virtualization technology is the backbone of cloud systems, the fastest-growing energy consumers, globally. As discussed in article [4], it holds an imperative position in the resource operation area by furnishing results for numerous affiliated critical problems and challenges. Virtual machine quilting, which is substantially addressed, allows all data centers to move from a particular state to a more optimized one. Presently, the unknown advancements in vessel-grounded pall and containerized workloads have created new connection openings. We give a general view on IT connection at different situations of cloud services.

Proposals that resource scheduling becomes the prominent issue in all computing due to the rapid-fire growth of on-demand requests and the miscellaneous nature of pall

Coffers. Pall provides energy, query, and plainness-grounded services to druggies in a pay-as-you-go fashion over the internet is mentioned in article [5]. In recent decades, an increase in requests (different and complex operations) for all services is raising the workload in the pall terrain. The cataloging algorithm should also optimize the crucial performance index parameters like response time, makespan time, trust ability, vacuity, energy consumption, cost, resource application, etc. To fulfill the below-mentioned ideal, numerous state of art scheduling algorithms have been proposed grounded upon heuristic, meta-heuristic, and mongrel, reported in the literature. This paper provides a methodical review as well as brackets of proposed scheduling ways along with their advantages and limitations.

With the increase of mega-cities, the demand for Smart metropolises is proliferating as discussed over the article [6]. The Mega-cities can be alter through the Cloud of effects (Hut). Effective energy consumption in Smart metropolises has a massive impact on the terrain. But, computational power is adding swiftly in the pall calculating terrain. Enormous energy consumption (EC) and Service position Agreement violation (SLAV) become pivotal concerns. The Virtual Machine (VM) connection approach can significantly reduce EC, and SLA violations (SLAV), and increase resource application. The trial shows that proposed algorithms reduce EC and SLAV in cloud data centers and can be used to construct a smart and sustainable terrain for Smart metropolises.

Energy consumption in cloud computing and an analysis is made which includes both the private and public clouds, article [7]. It includes energy consumption in switching and transmission as well as data processing and data storage. We can make more energy-efficient use of computing power in cloud computing and this is made especially when the computing tasks are of low intensity or infrequent.

Data centers are a source of environmental concerns because of their drastic energy consumption and high carbon emissions. Article [8] focuses on various elements that can reduce the negative impacts of the cloud. It is done after a deep analysis of cloud systems with respect to their power efficiency. It is found that the key driver technology for energy-efficient clouds is "Virtualization" which significantly improves the energy efficiency of cloud providers by leveraging the economies of scale associated with the large number of organizations sharing the same infrastructure. This paper also explains the four key factors-dynamic provisioning, multi-tenancy, server utilization, and data center efficiency that has enabled organizations to reduce their carbon emissions by 30% and thus help in the consumption of energy and protect the environment.

With the added growth of large data storehouses and computational demand, Green Cloud Computing is known to be a broad area and hot field for exploration is discussed in the article [9]. Therefore, there's a need for Green Cloud computing which can produce results that not only make the IT coffers energy effective but also minimize the functional costs. It includes a huge number of focus areas in order to give proper operation of power, virtualization of waiters, design of data centers, recovering styles, eco-labeling, terrain sustainability design, and energy-efficient coffers, etc. In this review, a brief discussion on Green Cloud computing is given also colorful operation areas of Green IT are bandied and reviewed further.

Cloud computing is a largely scalable and cost-effective structure for running HPC, enterprise, and Web operations, article [10]. In order to design similar results, deep analysis of Cloud is needed with respect to its power effectiveness. Cloud computing is a natural extension of virtualization technologies that enable the scalable operation of virtual machines over a plethora of physically connected systems. Therefore, we bandy colorful rudiments of shadows that contribute to the total energy consumption and how it's addressed in the literature. We also bandy the recrimination of these results for unborn exploration directions to enable green cloud computing.

3. Scheduling mechanisms in cloud computing

Pre-emptive and non-pre-emptive are terms used in operating systems to describe how the system handles the execution of processes or threads. Here are some notes on preemptive and non-pre-emptive scheduling:

1. **Non-pre-emptive scheduling**: In non-pre-emptive scheduling, a process or thread that is currently executing is not interrupted until it has completed its task or it voluntarily yields the CPU. The CPU is only given to other processes when the current process has finished executing. This approach is also known as cooperative scheduling since processes have to cooperate with each other to ensure that each process gets its fair share of CPU time.

   Examples of non-pre-emptive scheduling include the First Come First Serve scheduling algorithm and the Shortest Job First scheduling algorithm.

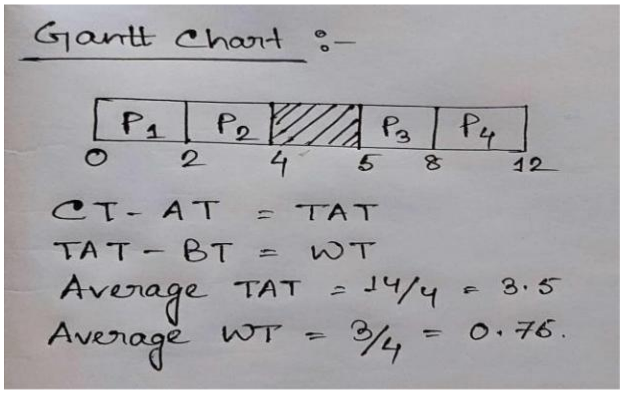
2. **Preemptive scheduling**: In preemptive scheduling, the operating system can interrupt a process or thread that is currently executing to give the CPU to another process. This means that a process can be interrupted in the middle of its execution, and its state is saved to allow it to resume later. This approach is also known as time-sharing since the operating system switches between processes quickly to give each process a fair share of CPU time.

   Examples of preemptive scheduling include the Round Robin scheduling algorithm and the Priority scheduling algorithm.

First Come First Serve scheduling: The First Come First Serve (FCFS) scheduling algorithm is a simple non-preemptive algorithm used in operating systems to allocate resources to processes. In FCFS scheduling, the process that arrives first is executed first, followed by the next process in the arrival order, and so on. This algorithm is easy to implement and straightforward, but it may not be the most efficient algorithm in terms of average waiting time or turnaround time. Despite its limitations, FCFS scheduling has its applications, such as in batch processing systems or when the length of the processes is known beforehand and their priority is not a concern. A first come first serve scheduling algorithm is shown in table in 3.1.

Processes	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P1	**0**	**2**	**2**	**2**	**0**	**0**
P2	**1**	**2**	**4**	**3**	**1**	**1**
P3	**5**	**3**	**8**	**3**	**0**	**0**
P4	**6**	**4**	**12**	**6**	**2**	**2**

Table No: 3.1 First Come First Serve Scheduling



FigNo : 3.1 Gantt Chart of First Come First Serve

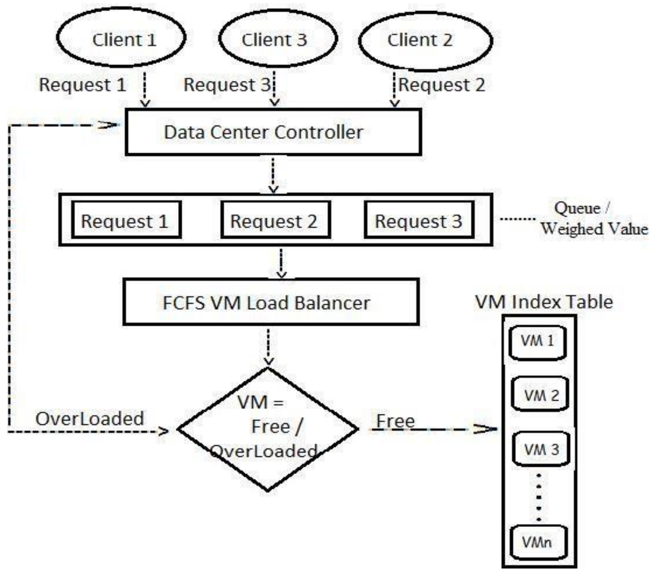
In FCFS scheduling, the processes are executed in the order in which they arrive. The process that arrives first is executed first, followed by the next process in the arrival order, and so on.

Using this information from fig.3.1, the scheduling algorithm works as follows:

1. At time t=0, P1 arrives and starts executing.
2. P1 runs for 2 units, until t=2.
3. At t=1, P2 arrives and is added to the ready queue. However, since P1 is still executing, P2 has to wait.
4. P1 completes its execution at t=2 and is removed from the system. P2 starts executing immediately after P1, at t=2.
5. P2 runs for 2 units, until t=4.
6. t=4 to t=5 no processes arrive for execution.
7. At t=5, P3 arrives and is added to the ready queue and starts work for execution.
8. P3 runs for 3 units, until t=8.
9. At t = 6, P4 arrives , P3 is still executing, P4 has to wait.

10. At t=12, all processes have completed their execution.

And that's it! The processes have completed their execution using the FCFS scheduling algorithm. The algorithm executes the processes in the order in which they arrive, ensuring that the process that arrives first is executed first.



Flow Chart No. 4:1 First Come First Serve Scheduling [14]

Shortest Job First scheduling: The Shortest Job First (SJF) scheduling algorithm is a non-preemptive scheduling algorithm used in operating systems to allocate resources to processes. In SJF scheduling, the process with the shortest expected processing time is executed first. If two or more processes have the same processing time, then the process that arrived first is executed first. SJF scheduling results in a shorter average waiting time and turnaround time compared to First Come First Serve (FCFS) scheduling, as it prioritizes the shortest jobs first. However, SJF scheduling may not be the most efficient algorithm if the processing time of the processes is not known beforehand or if there is a risk of starvation for longer processes. A shortest job first scheduling algorithm is shown in table 3.2.

Example:

Processes	**Arrival Time**	**Burst Time**	**Completion Time**	**TAT**	**WT**	**RT**

P1	**1**	**3**	**6**	**5**	**2**	**2**
P2	**2**	**4**	**10**	**8**	**4**	**4**
P3	**1**	**2**	**3**	**9**	**0**	**0**
P4	**4**	**4**	**14**	**10**	**6**	**6**

Table No: 3.2 Shortest Job First scheduling

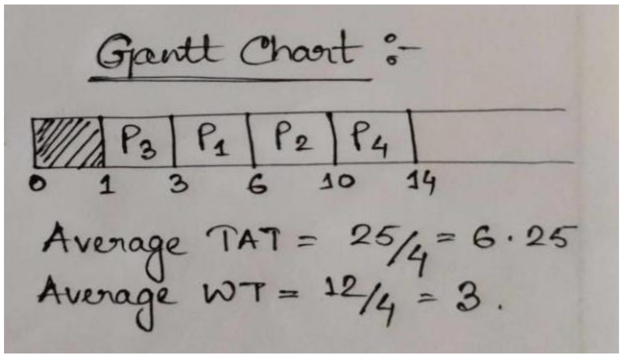


Fig No: 3.2 Gantt chart of Shortest Job First Scheduling

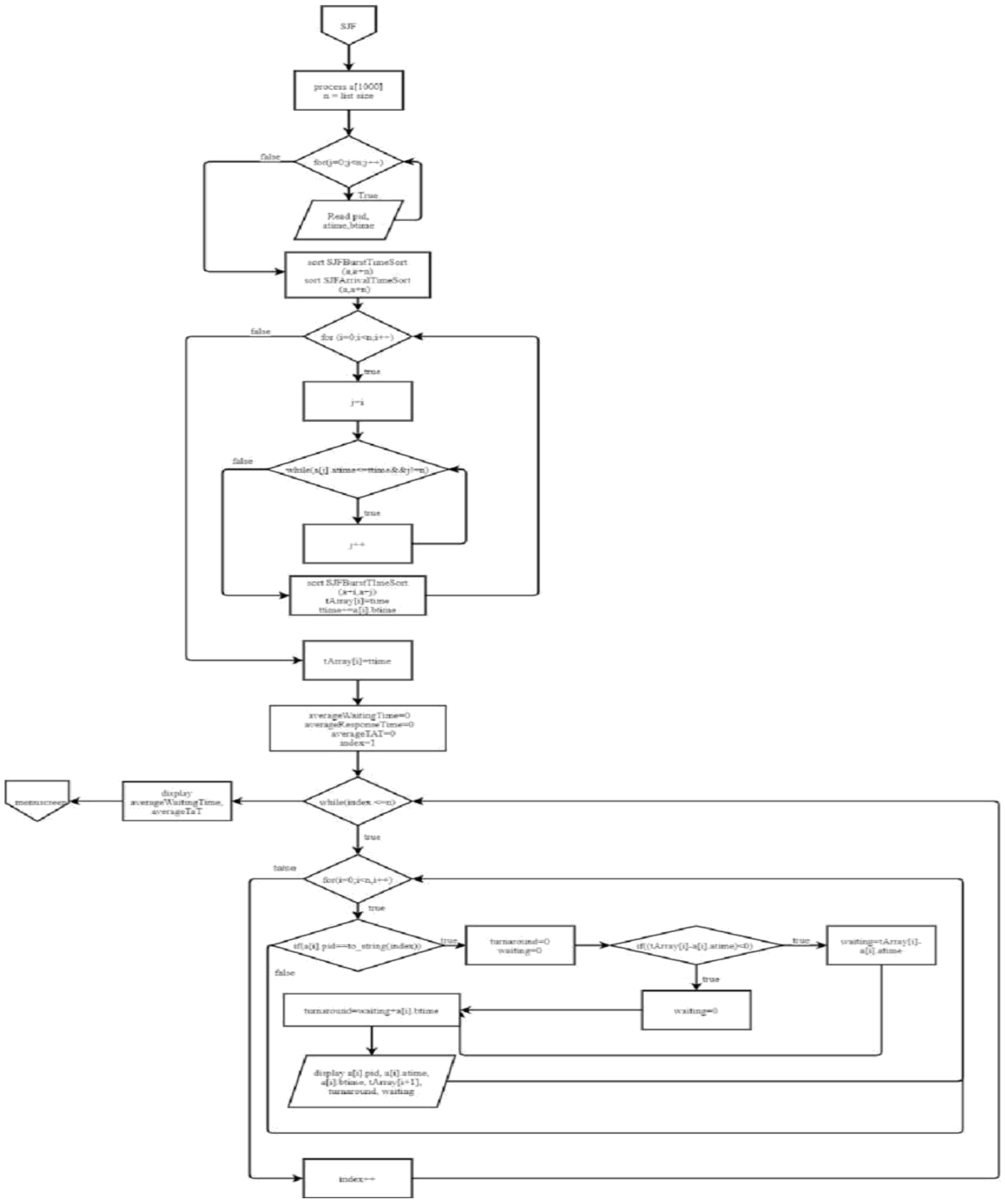
In SJF scheduling, the processes are executed in the order of their burst times, with the shortest job executed first. If two or more processes have the same burst time, then the process that arrived first is executed first.

Using this information from fig.3.2, the scheduling algorithm works as follows:

1. At time t=1, P1,P3 arrives, added to the ready queue.
2. Starts executing according to their lowest Burst Time . Hence P3 Burst Time is lower than P1, So P3 runs first and runs for 2 units, Until t = 3.
3. at t = 2 , P2 arrives and adds to the ready queue. Now P1 and P2 have to wait for execution.
4. Hence P1 comes first , it runs first at t=3, P2 starts for execution for 3 units, until t=6 .
5. At the t =4 P4 comes to the ready queue for execution but has to for the P1 and P2 executions.
6. Now, P2 and P4 are present in the ready queue. Hence P2 comes first ,P2 starts for execution for 4 units, until t = 10. P4 has to wait.
7. At t = 10 P4 starts to execute for 4 units until t=14 is removed from the system.

8. At t=14 all processes have completed their execution.

And that's it! The processes have completed their execution using the SJF scheduling algorithm. The algorithm executes the processes in the order of their burst times, with the shortest job executed first. This results in a shorter average waiting time and turnaround time compared to FCFS scheduling.



Flow Chart No. 4:2 SJF Implementation [15]

Round-robin scheduling: This mechanism allocates resources to tasks or applications in a cyclical manner. Each task or application is given a turn to use the resources for a fixed amount of time before the next task or application is given a turn. A round-robin scheduling algorithm showing the arrival and burst time is shown in table 3.3.

Processes	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P1	**0**	**5**	**12**	**12**	**7**	**0**
P2	**1**	**4**	**11**	**10**	**6**	**1**
P3	**2**	**2**	**6**	**4**	**2**	**2**
P4	**4**	**1**	**9**	**5**	**4**	**4**

Table No: 3.3 Round-robin scheduling

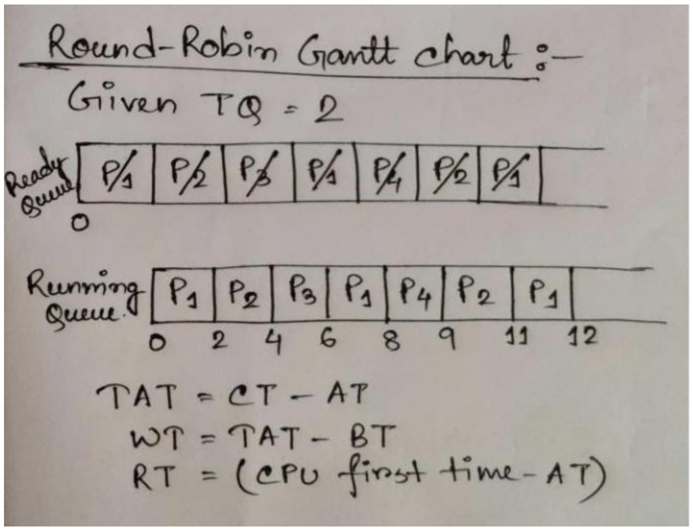


Fig no.3.3 Gantt Chart of Round Robin Scheduling Algorithm

From fig 3.3,If we use a time quantum of 3 units, the scheduling algorithm works as follows:

1. At time t=0, P1 is added to the ready queue since it has arrived.
2. P1 starts executing at time t=0 and runs for 2 units, until t=2.
3. At t=2, P1 is preempted and added back to the ready queue.
4. P2 is added to the ready queue since it has arrived and is the next process in the queue.

5. P2 starts executing at t=2 and runs for 2 units, until t=4.
6. At t=4, P2 is preempted and added back to the ready queue.
7. P3 is added to the ready queue since it has arrived and is the next process in the queue.
8. P3 starts executing at t=4 and runs for 2 units, until t=6 , completes the execution and is removed from the system.
9. P1 is the next process in the queue, so it starts executing again at t=6 and runs for 2 units, until t=8.
10. At t=8, P1 is preempted and added back to the ready queue.
11. P4 is the next process in the queue, so it starts executing again at t=8 and runs for the remaining 1 unit, until t=9.
12. At t=9 starts executing for the remaining 2 units , until t = 11 , completes the execution and is removed from the system.
13. At t=11, P1 starts executing for the remaining 1 unit , until t = 12 .
14. At t=12 all processes have completed their execution.

And that's it! The processes have completed their execution using round-robin scheduling with a time quantum of 3 units.

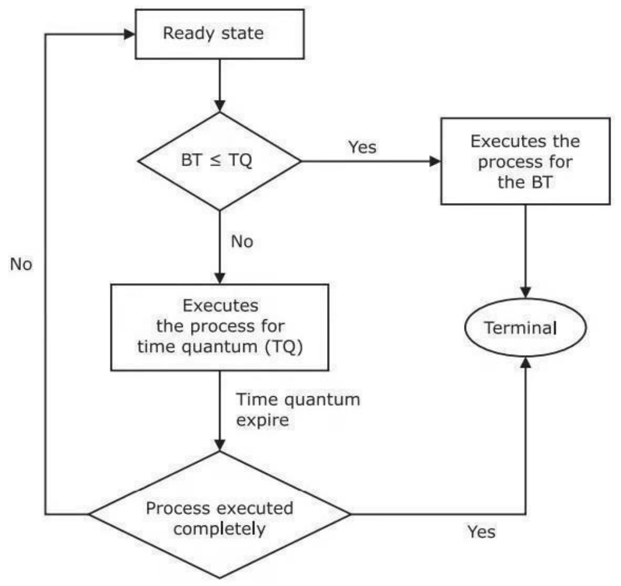

Ready state
BT ≤ TQ
Yes
Executes the process for the BT
No
No
Executes the process for time quantum (TQ)
Terminal
Time quantum expire
Process executed completely
Yes

Fig No. 3.4 (b): Round Robin Implementation [16]

Priority scheduling: This mechanism prioritizes tasks or applications based on their importance or urgency. Higher priority tasks or applications are given access to resources before lower priority ones. A priority scheduling algorithm is shown in table 3.4.

Process	**Arrival Time**	**Burst Time**	**Priority**	**Completion Time**	**TAT**	**WT**
P1	**0**	**5**	**10**	**12**	**12**	**7**
P2	**1**	**4**	**20**	**8**	**7**	**3**
P3	**2**	**2**	**30**	**4**	**2**	**0**
P4	**4**	**1**	**40**	**5**	**1**	**0**

Table No: 3.4 Priority scheduling

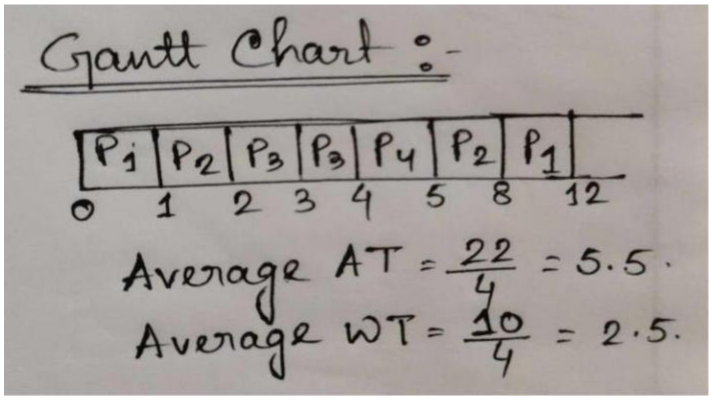


Fig no.3.4: Gantt Chart of Priority Scheduling Algorithm

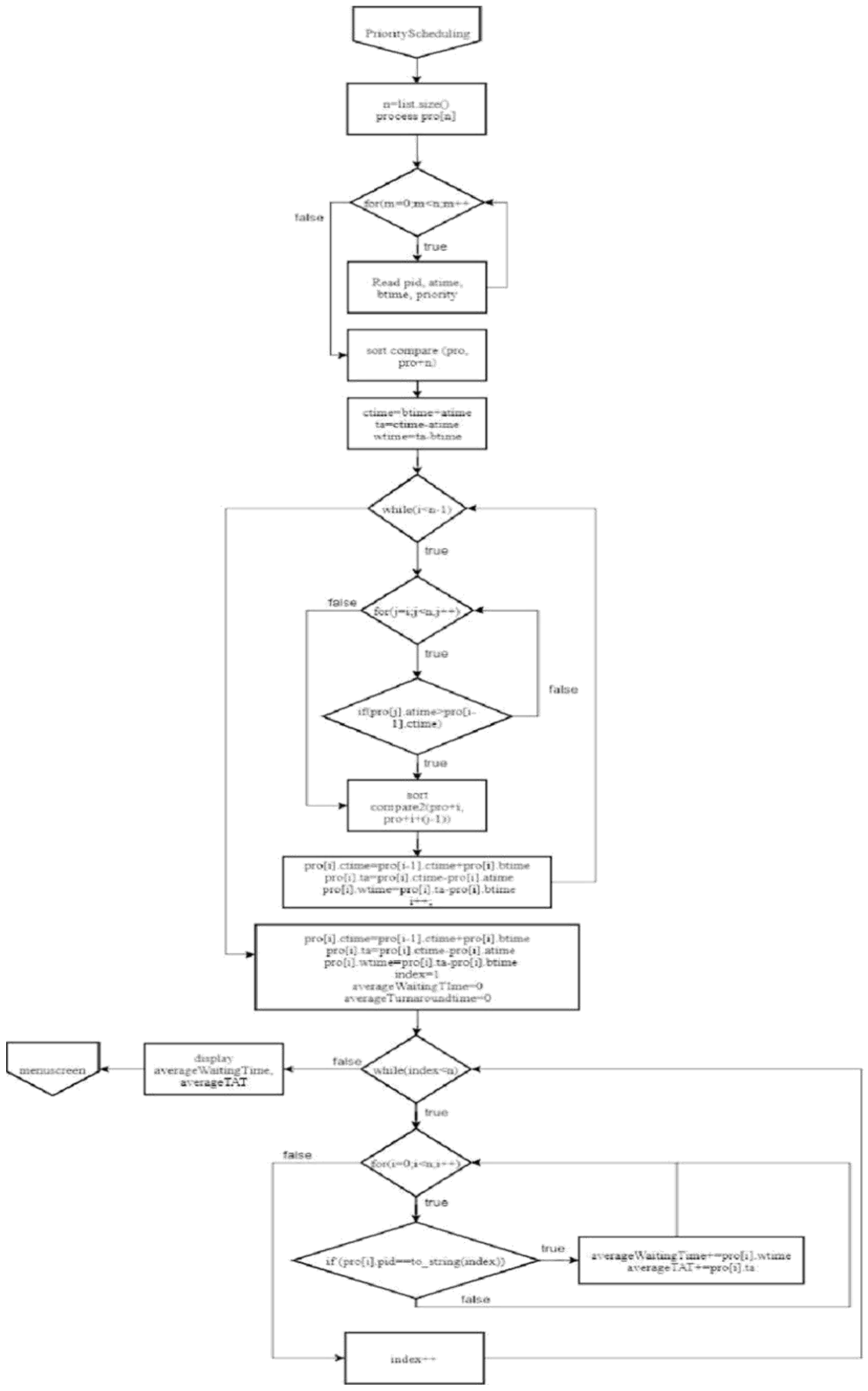
In priority scheduling, the process with the highest priority is selected for execution first. If multiple processes have the same priority, then the process that arrived first is selected.

Using this information, the scheduling algorithm works as follows:

1. At time t=0, P1 is the only process in the ready queue and starts executing.
2. At t=1, P2 arrives and is added to the ready queue. Since P2 has a higher priority than P1, P2 is selected to execute next.
3. P2 starts executing at t=1 and runs for 1 unit, until t=2.
4. At t=2, P3 arrives and is added to the ready queue. Since P3 is a higher priority, P3 is selected to execute next for 1 unit. At t=4,P3 completes execution and is removed from the system.

5. at t=4 P4 arrives . Since P2 is a higher priority P4 is selected to execute and it executes for 1 unit . At = 5, P4 completes execution and is removed from the system.
6. Since P2 has higher priority than P1 ,P2 is selected for remaining execution work.
7. At t=8 P2 completes execution and is removed from the system.
8. P1 resumes execution at t=8 and runs for the remaining 4 units, until t=12.
9. At t=12, all processes have completed their execution.

At t=12, all processes have completed their execution.And that's it! The processes have completed their execution using priority scheduling. The algorithm selects the processes with the highest priority for execution first, and in case of a tie, the process that arrived first is selected.


PriorityScheduling
n=list.size()
process pro[n]
for(m=0;m<n;m++
false
true
Read pid, atime, btime, priority
sort compare (pro, pro+n)
ctime=btime+atime
ta=ctime-atime
wtime=ta-btime
while(i<n-1)
true
false
for(j=i;j<n,j++)
true
false
if(pro[j].atime>pro[i-1].ctime)
true
sort compare2(pro+i, pro+i+(j-1))
pro[i].ctime=pro[i-1].ctime+pro[i].btime
pro[i].ta=pro[i].ctime-pro[i].atime
pro[i].wtime=pro[i].ta-pro[i].btime
i++;
pro[i].ctime=pro[i-1].ctime+pro[i].btime
pro[i].ta=pro[i].ctime-pro[i].atime
pro[i].wtime=pro[i].ta-pro[i].btime
index=1
averageWaitingTIme=0
averageTurnaroundtime=0
menuscreen
display averageWaitingTime, averageTAT
false
while(index<n)
true
false
for(i=0;i<n;i++)
true
if (pro[i].pid==to_string(index))
true
averageWaitingTime+=pro[i].wtime
averageTAT+=pro[i].ta
false
index++

Flow Chart No. 4.4 Priority Scheduling [17]

3.1 Some different terminologies are

1. **Time Quantum (TQ):** The time quantum specifies the maximum amount of time that a process is allowed to run on the CPU before it is preempted and added back to the end of the ready queue. The time quantum is denoted by TQ.
2. **Arrival Time (AT):** The arrival time of a process is the time at which the process arrives in the system and is added to the ready queue. The arrival time is denoted by AT.
3. **Burst Time (BT):** The burst time of a process is the amount of time required by the process to complete its execution on the CPU. The burst time is denoted by BT.
4. **Completion Time (CT):** The completion time of a process is the time at which the process completes its execution on the CPU. The completion time is denoted by CT.
5. **Turnaround Time (TAT):** The turnaround time of a process is the time interval between the arrival of the process and its completion on the CPU. The turnaround time is denoted by TAT and is calculated as TAT = CT - AT.
6. **Waiting Time (WT)**: The waiting time of a process is the time interval between the arrival of the process and the start of its execution on the CPU. The waiting time is denoted by WT and is calculated as WT = TAT - BT.

4. Motivation

If there are n number of servers and n number of users, the ratio will be 1:1 to provide the most effective service, but it will not be efficient in terms of resource utilization like bandwidth, and electricity. So our motivation is to find a way to provide the highest service using optimum utilization of resources using minimal servers including the backup.

5. Problem Identification

The use of cloud computing is increasing all over the world. They are used to provide on-demand network access to a shared pool of computing resources including networks, servers, storage, applications and services that can be deployed with minimum effort and service provider interactions. But, cloud data centers consume huge amounts of power and electricity, which in turn produces shortages in energy and global climate.

In Round Robin scheduling, the CPU executes a process for a fixed time quantum (TQ) and then switches to the next process in the ready queue. The process that is preempted is added back to the end of the ready queue. The Round Robin scheduling is as follows:

1. Set the current time t = 0.
2. While there are still processes in the ready queue:
    - Choose the first process in the ready queue.
    - If the burst time of the process is less than or equal to the time quantum TQ, execute the process for the entire burst time and calculate its completion time.
    - If the burst time of the process is greater than the time quantum TQ, execute the process for TQ units of time and then preempt it and add it back to the end of the ready queue. Calculate the completion time as the current time plus TQ.
    - Calculate the turnaround time and waiting time for the process and move it to the completed queue.
    - Increment the current time by the burst time or the time quantum, whichever is smaller.
3. Calculate the average turnaround time and average waiting time for all completed processes.

6. Proposed Solution

The Round Robin scheduling algorithm pseudo-code is discussed below:

```
1. Set time quantum, q (maximum amount of time a process can run on the CPU).
2. Initialize a queue to hold the processes in the ready state.
3. Set the current time, t,
to 0. while processes are in
the ready queue:
    select the first process in the queue
    if the process's remaining time is less than or equal to the
        time quantum: execute the process for its remaining time
        update the process's wait time and
        turnaround time remove the process from
        the ready queue
    else:
        execute the process for the time
        quantum update the process's
        remaining time
        move the process to the end of the ready queue
```

This is an implementation of the Round Robin scheduling algorithm, and there are several variations and optimizations that can be made depending on the specific requirements and constraints of the system.

7. Experimental Results

The platform used for stimulation - PyCharm .

PyCharm is an integrated development environment (IDE) used for Python programming.

System Processor : Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80

GHz Installed Ram : 8.00 GB (7.85 GB usable)

System Type: 64-bit operating system, x64-based processor

The method used for the purpose of stimulation - Round Robin Scheduling Algorithm.

Input:

Enter the number of processes: 10

Process No.	Arrival Time	Burst
1	1	4
2	2	5
3	0	10
4	2	5
5	4	6
6	3	7
7	5	9
8	3	8
9	7	12
10	1	12

Table No : 7.1 Input Table

First, the number of processes are provided by the user and then provided arrival time and burst time of each process.

take input as per the above mentioned in table no.7.1.

Output (Result)

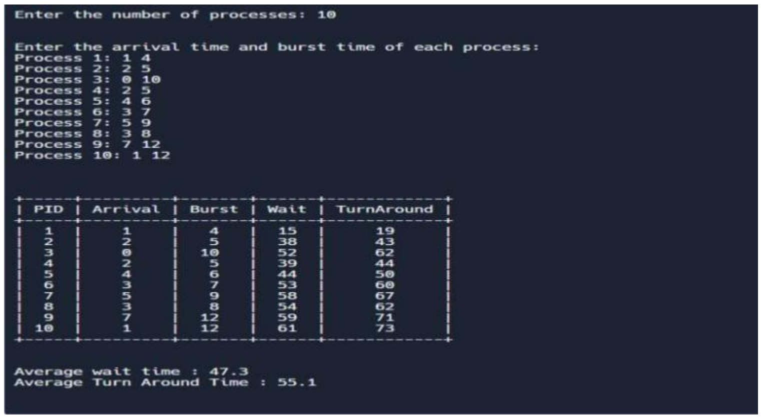
The output table is shown below in table no. 7.1 -

PID	Arrival	Burst	Wait	Turn Around
1	1	4	15	19
2	2	5	38	43
3	0	10	52	62
4	2	5	39	44
5	4	6	44	50
6	3	7	53	60
7	5	9	58	67
8	3	8	54	62
9	7	12	59	71
10	1	12	61	73

Table No : 7.2 Output Table

Average wait time: 47.3

Average Turn Around Time: 55.1



```
Enter the number of processes: 10

Enter the arrival time and burst time of each process:
Process 1: 1 4
Process 2: 2 5
Process 3: 0 10
Process 4: 2 5
Process 5: 4 6
Process 6: 3 7
Process 7: 5 9
Process 8: 3 8
Process 9: 7 12
Process 10: 1 12


+-----+---------+-------+------+------------+
| PID | Arrival | Burst | Wait | TurnAround |
+-----+---------+-------+------+------------+
|  1  |    1    |   4   |  15  |     19     |
|  2  |    2    |   5   |  38  |     43     |
|  3  |    0    |  10   |  52  |     62     |
|  4  |    2    |   5   |  39  |     44     |
|  5  |    4    |   6   |  44  |     50     |
|  6  |    3    |   7   |  53  |     60     |
|  7  |    5    |   9   |  58  |     67     |
|  8  |    3    |   8   |  54  |     62     |
|  9  |    7    |  12   |  59  |     71     |
| 10  |    1    |  12   |  61  |     73     |
+-----+---------+-------+------+------------+


Average wait time : 47.3
Average Turn Around Time : 55.1
```

Fig No: 7.2 Screenshot of the output

The image fig no: 7.2 showing the waiting and turnaround time of each output is shown above.

We get the output as per the above mentioned in table no.7.2.The waiting time(wait) and the turnaround time(Turn Around) are calculated accordingly for each process. The average of waiting and turnaround time are also calculated.

8. Conclusion

Virtualization technologies has given rise to the use of cloud computing as business models can access distributed services like IaaS, SaaS, PaaS in a pay-as-you-use model based on their usage. Virtualization enables the same physical machine to host and run multiple virtual servers for better resource utilization but cloud infrastructures face a number of challenges like resource management, power consumption, cost and performance. Data centers deployments have the major inefficiencies due to the poor management of resources .Therefore, a green-cloud computing architecture is necessary for the energy evaluation of these data centers. Instead of using one server can for one user, in this project we have tried to reduce the number of servers but provide the optimum service by scheduling the servers using round robin scheduling algorithm so that different number of servers can be allocated to address the requests of the users.

The solution provided in this project has been done to provide the maximum output to the users keeping in mind to reduce the negative impacts caused by cloud computing.

The result from this experiment is that each request to the process has been allocated and completed within the given time frame.

The project has been done to solve the problem but there is always scope for further modifications in future and help in further problems. This project can be further improved by increasing the number of servers and other such changes as per the requirement of the users.

9. Reference

[1].Mohit Kumar, S.C. Sharma, Anubhav Goel, S.P. Singh.A comprehensive survey for scheduling techniques in cloud computing.Journal of Network and Computer Applications(2019) 1-33.

[2]. Li Mao, Yin Li, Gaofeng Peng , Xiyao Xu, Weiwei Lin.A multi-resource task scheduling algorithm for energy-performance tradeoffs in green clouds.Sustainable Computing: Informatics and Systems(2018),233-241.

[3]. Leila Helali, Mohamed Nazih. A survey of data center consolidation in cloud computing systems. Computer Science Review(2021)1-28.

[4].Saikin Long, Yuan Li , Jinna Huang , Zhetao Li , Yanchun Li.A review of energy efficiency evaluation technologies in cloud data centers.Energy & Buildings(2022)1-14.

[5]. Nirmal Kr. Biswas, Sourav Banerjee, Utpal Biswas , Uttam Ghosh.An approach towards development of new linear regression prediction model for reduced energy consumption and SLA violation in the domain of green cloud computing.Sustainable Energy Technologies and Assessments(2021)1-9.

[6]. Ayaz Ali Khan, Muhammad Zakarya. Review article Energy, performance and cost-efficient cloud data centres: A survey.Computer Science Review(2021)1-31.

[7].Jayant Baliga, Robert W. A. Ayre, Kerry Hinton, and Rodney S. Tucker. Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport" Review For processing large amounts of data, management and switching of communications may contribute significantly to energy consumption and cloud computing seems to be an alternative to office-based computing,IEEE(2010)149-165.

[8]. Saurabh Kumar Garg and Rajkumar Buyya.Green Cloud computing and Environmental Sustainability.Harnessing Green It:Principles and Practices(2012)1-27.

[9]. Jianxin Li, Bo Li, Tianyu Wo, Chunming Hu, Jinpeng Huai, Lu Liu, KP Lam. CyberGuarder: A Virtualization security assurance architecture for Green Cloud.Future Generation Computer Systems(2012)379-390.

[10]. Yashwant Singh Patel, Neetesh Mehrotra, Swapnil Soner.Green Cloud Computing: A review on Green IT areas for cloud computing environment.IEEE.(2015)1-4.

[11]. Url : https://www.javatpoint.com/cloud-computing-architecture
[12]. Url : https://www.javatpoint.com/types-of-cloud

[13]. Url : https://www.stackscale.com/blog/cloud-service-models/

[14]. Url : https://www.researchgate.net/figure/Flowchart-of-FCFS-Algorithm_fig2_30164 8483

[15]. Url : https://www.researchgate.net/figure/Flow-chart-for-SJF-implementation_fig3_353120834
[16]. Url : https://byjusexamprep.com/round-robin-scheduling-algorithm

i?lang=en

[17]. Url: https://www.researchgate.net/figure/Flow-chart-for-priority-

scheduling implementation_fig4_353120834

10. Appendix

Implementation in Code

```
# implementation code in python 3 using round robin algorithm
class Process:
    def_init_(self, pid, arrival: int, burst: int):
        self.pid = pid #pid = Process id (number of process)
        self.arrival = arrival # process arrival time
        self.burst = burst # time required to finish work
        self.remaining = burst #remaining time to finish the process
        self.completion = 0 #initially completion time is 0
        self.turnaround = 0 # turnaround time = completion – arrival . initially 0
        self.waiting = 0 # waiting time = turnaround – burst . initially 0

quantum = int(input("Enter time quantum: "))#variable quantum = (enter the time
quantum (gap between the process)by user)
n = int(input("Enter the number of processes: "))#take variable n and enter the
number of processes by user .
ProcessList: list[Process] = []#process list
Running: list[Process] = []#running processes
Finished: list[Process] = []#finished list of processes
time: int = 0

print("\n\nEnter the arrival time and burst time of each process:") #enter each
process arrival time and burst time.
for i in range(n):
    a, b = list(map(int, input(f"Process {i+1}: ").split()))#using map function
for double_even()
    ProcessList.append(Process(i+1, a, b))

ProcessList.sort(key=lambda  p:  p.arrival) #arrival  time  sorting

while ProcessList or Running:

    if Running:
        p = Running.pop(0) #pop from the running and store in p variable
        t = min(p.remaining, quantum) #t = minimum between remaining process and
quantum
        time += t
        p.remaining -= t  #t = minimum between remaining process and
quantum

        while ProcessList:
            if ProcessList[0].arrival <= time: #while arrival time of process list index 0
is smaller equal to time then pop the arrival time from the process list and add to the running list.
                Running.append(ProcessList.pop(0))
            else:
                break #conditions false break the process.
#calculation part of waiting time , completion time , turnaround time

        if p.remaining == 0:
            p.completion = time
            p.waiting = p.completion - p.arrival - p.burst
            p.turnaround = p.waiting + p.burst
            Finished.append(p)
        else:
            Running.append(p)
    else:
        time = ProcessList[0].arrival
        Running.append(ProcessList.pop(0))

Finished.sort(key=lambda p: p.pid)

# print output

print("\n" * 2)
print("+-------+-------------+----------+---------+-----------------+")
print("| PID | Arrival | Burst | Wait | TurnAround |")
print("+-------+-------------+----------+---------+-----------------+")
for p in Finished:
    print(f"|{p.pid: ^5}|{p.arrival: ^9}|{p.burst: ^7}|{p.waiting:
^6}|{p.turnaround: ^12}|")
print("+-------+-------------+----------+---------+-----------------+")

print("\n\nAverage wait time :", sum(p.waiting for p in Finished)/len(Finished))
print("Average Turn Around Time :", sum(p.turnaround for p in
Finished)/len(Finished))

input("\n\nPress Enter to Exit..")
      #process exits
```