Dynamic Resource Allocation

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Shubham Kumar (191358)

Paritosh Sengar(191375)

Under the supervision of

Dr. Monika Bharti (Assistant Professor, CSE)

.

Department of Computer Science & Engineering and Information Technology

Candidate's Declaration

I hereby declare that the work presented in this report entitled "Dynamic Resource Allocation " in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2023 under the supervision of (Dr. Monika Bharti) (Assistant Professor, CSE).I also authenticate that I have carried out the above mentioned project work under the proficiency stream,Cloud Computing. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shubham Kumar 191358

Paritosh Sengar 191375

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Supervisor Name:Dr Monika Bharti

Designation:Assistant Professor

Department name:CSE

Dated:26/04/2023

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ____________________________ __**Department:** _________________ **Enrolment No** _________

Contact No. _______________________________**E-mail.** ______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Knowledge, energy, and time are the resources in the completion of this project but the most requisite is the proper guidance. We would like to express our deepest appreciation to all those who have been helping us throughout the project and without whom this project would have been a very difficult task. We are highly indebted to our respected project guide Dr. Monika Bharti for his valuable suggestions, ideas, and constant supervision as well as for providing necessary information throughout the project. We want to express gratitude towards Prof. Dr. Vivek Kumar Sehgal, Head of the Department, Department of CSE, Jaypee University of Information Technology, Waknaghat for providing us this opportunity to make our project. We would also like to thank all the panel members for their valuable suggestions in all the viva-voice during evaluation from which we got to learn a lot. No thanks can counter our indebtedness to our parents who have been with us throughout. We thank them from the core of our hearts.

Date: 26/04/23

Table of Contents

LIST OF ACRONYMS AND ABBREVIATIONS

CC-Cloud Computing

Cloudsim:CloudSim is a framework for modeling and simulation

Task Scheduling Algorithms

VM-Virtual machine

Algo-Algortithm

Web App-Website based Application

LIST OF FIGURES

ABSTRACT

Cloud computing manage a variety of virtualized resources which makes scheduling a critical component. In the cloud , a client may utliize several thousand virtualized assets for every task. Consequently, a manual scheduling is not a feasible solution. The basic idea behind task scheduling is to slate tasks for the minimization of time loss and maximize performance . Several research efforts have examined task scheduling in the past. This project presents a comprehensive survey of task scheduling strategies and the associated metrics suitable for cloud computing environments. Distinctive scheduling procedures are studied to discover which characteristics are to be included in a given system and which ones to be disregard.The project presents a review study of various task scheduling algorithms in cloud environment including RR,MaxMin,MinMin,FCFS,Selective,SJF.The algorithmhas been tested using CloudSim toolkit. It is recommended to use the algorithms with best make span time ,average waiting time,completion cost in order to avoid starvation and making it suitable for load balancing.

Keywords : CloudSim, FCFS, Genetic Algorithm, Round Robin Scheduling Algorithm, Virtual Machine Scheduling,SJF Algorithm,Maxmin Algorithm,Minmin Algorithm,Selective Algorithm.

Chapter 1 Introduction

1.1 Introduction

The term "Cloud" refers to network of linked computers that contains multiple unified computing resources. The development of cloud computing in recent years has made it possible to quickly assemble a network of geographically scattered data centres for the provision of high-quality and reliable services . Cloud computing is now a successful paradigm for providing computational capabilities on a "pay-per-utilize" basis.The IT industry is changing and becoming more uniform thanks to cloud computing. Cloud computing has many opportunities for growth while also posing numerous challenges to the evolution of traditional IT due to its expanding application and promotion . Cloud computing has recently become a popular IT(internet) strategy empowering users. It provide access to common reservoir Configurable resources that are available immediately and can be released with minimum management or help from cloud providers . This innovation has a number of benefits, including the ability to enhance time, cost, stack adjusting, and storage benefits in the marketplace. With the help of this invention, all applications can continue to operate on a virtual platform, and all resources are divided the VM . Each and application is unique independent.

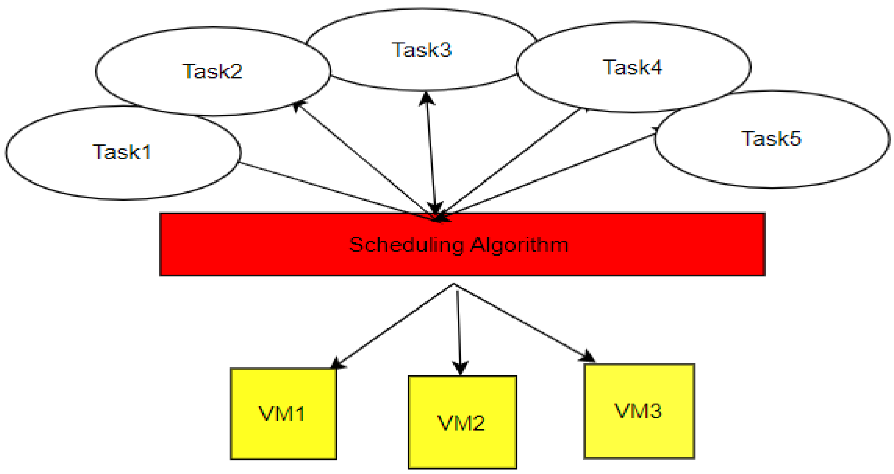


Fig 1.1 Scheduling Algorithms

Some parallel programmes display a decrease in CPU resource utilisation when parallelism is increased. If the jobs are not scheduled appropriately, performance degrades due to the cloud operations' enormous volume of data. Thus, in the sphere of cloud computing, the scheduling method is crucial.To plan and carry out the tasks with the highest assessed gain or advantages, a scheduling algorithm is used.Users use tens of thousands of resources, making it hard to assign each work by hand. The VM layers handle the scheduling tasks.In cloud computing, scheduling is crucial for allocating resources to each activity effectively and efficiently.

1.1.1 VM Scheduling

Processes or activities are planned according to the provided requirements and employed methods in a balanced scenario called scheduling. Cloud computing,Virtual Machine scheduling algoritthms used for scheduling , Virtual servers requests to the actual machine of specific data centers in accordance with the needs satisfied by the requested resources. Schdeuling Algorithms works on the following rules.

- For VM find PM.
- Determine the allocation for VM
- Schedule the jobs for VM.

As was mentioned in the introduction section, a significant component of managing cloud resources cost-, energy-, and security-wise is cloud deployment. Three phases make up cloud provisioning: provisioning virtual machines, provisioning resources, and provisioning applications [7]. Here, provisioning for virtual machines must be the main concern.

Cloud deployment is also a key factor for managing resources in a costeffective,energyefficient, cloud provision consists of 3 step: (i) CM provision, (ii) resource (iii) APP provision. Here we need to focus on VM Allocation.

Task SchedulingAlgorithmsCategories:

- Instant Scheduling-A new task arrives, these scheduled directly on the VMs.
- BatchSchedulingTasks is combined batches before submission. These types are also called a mapping event.
- Static scheduling- regarded as being considerably simpler than dynamic scheduling. This is based on prior knowledge about the system's general condition.
- Dynamic Scheduling- Jobs are distributed according to the capacity of all the available VMs, taking into consideration the VM(s) current states and without knowing the system's overall condition beforehand.
- Preemptive scheduling-Each of the task can be interrupted while execution and resource to complete execution.
- NonPreemptiveScheduling-VM are not assign new tasks are scheduled tasks have finished running.

1.1.3 Task Scheduling Method

- A task is sent to a scheduler by a CC user. For resource information, a scheduler communicates with the cloud information system.
- The scheduler received the resource information from CIS.
- The scheduling algorithm performs its function by assigning tasks to the appropriate resources and submitting them to the successful resources.
- Through a cloud interface, the user obtains the resource's identifier and makes use of it.
- According to the timetable, the user delivers the resources.
- To control schedule scheduler receives periodically update the information the cloud condition.

1.1.4 Task Scheduling system in Cloud Computing

- There are 3 tiers in the cloud computing work scheduling system.
- A group of tasks send the cloud users that are necessary for exec. make up the first task level.
- The second scheduling level is in charge of assigning tasks to the appropriate resources to achieve the best resource utilization with the shortest lead time. The makespan time is the total time required complete the tasks from beginning end.
- The tasks shown in the accompanying figure are carried out at the third level by a group of people.

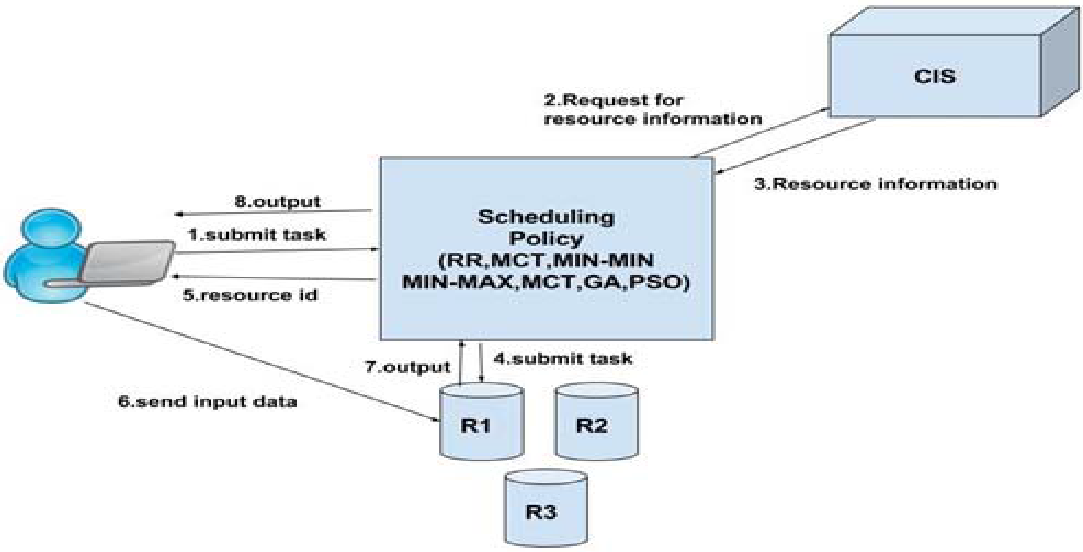


Fig 1.2 Task scheduling System

1.1.5 Types of Task Scheduling

I. Dynamic Task scheduling

The preceding information about framework applications and resources are not required for the dynamic load balancing.Shift the process from highload machines to low-load machines. Supports process slicing . There are various wellknown algos used in CC environments.

II. Static Load balancing

It needs previously specified details framework app,resources. A load balancing algorithm is referred to as static load balancing. distributes the backlog according to a set of fixed rules that are identified by characteristics of the currently active tasks.

III. Task scheduling algorithms advantages

- Control the QoS and performance of cloud computing.
- Control the CPU and the memory. .
- Improving the fairness of all the tasks.
- Scheduling tasks in real time basis.
- Improving load balancing.
- Achieving a high system throughput.
- Completed the tasks on time .

1.1.6 Scheduling Algorithms

I RR algortihm

The RR algo mainly idea of time slice . Time is divided several quantums. Each node receives a specific time Quantum, during the time, node performs all tasks . The RR is based sampling. Select loads in random manner where some servers are lightly loaded or some are heavy loads.

Pros:

- The suggested VM can be retrieved without any preprocessing steps.
- Equally distribute the load among the VMs.
- emphasizes equity among the assigned duties.
- Jobs are carried out sequentially, without ever waiting for the completion of a preceding job (starvation free).
- The scheduler won't wait for a VM to run out of processing power before moving on to the next one.
- It is founded on a basic principle.

Cons:

- Long tasks require extra time to execute.
- overloading
- Policies slicing are dependent on the size of the time slice, and in the case slice, many switching occur.

II. SJF Algorithm :

The SJF policy selects jobs that require shortest process time first. Shorter jobs are then run first. The main problem with SJF is the need to calculate the processing time for each job. If two jobs have the same processing time, SJF uses fcfs policy. This algo. also known as the shortest cpu burst time algo.

Pros :

- Decrease the average wait time since it decreases the wait time for quick jobs.

- increase the avg. waiting time of heavy tasks.

Cons:

- Starvation for heavy tasks

III. FCFS:

FCFS algorithm is simplest scheduling algorithm. When job come the less executiontime job is done fist.FIFO queues are used to implement FCFS policies.

Pros:

- Simple
- Easy to understand

Cons:

- Nonpreemptive.
- Less execution job is done so starvation.

IV. Min-Min Algorithm:

The Min-min method ensures that each task is completed in the shortest amount of time by allocating each task to a virtual computer known as a resource. Determine the duration of each job's execution on each resource that is available. There are two steps in the Minmin algorithm. The work with the quickest execution time is determined in first stage. Later on, the job with the fastest execution time is chosen. Then, algorithms assign resources to projects. Up till all jobs are mapped, this process is repeated.

Pros:

- Less makespan.
- No pre-checking for machines Increase the throughput.

Cons:

- The heavy tasks have to wait , Unbalanced load

V. Maxmin Algorithm

The maxmin method maps each job to the resource VM in such a way that each job can be finished in reasonable amount of time. Every job's execution time on each resource that is available is calculated. The assignment that requires the most time to complete is then chosen. The maxmin algorithms require less waiting time for large jobs.

Pros:

- Avg waiting less.
- Util. Increases.
- Min response time.

Cons:

- selects big tasks.
- lead to starvation.

VII. Selective Algorthim

We developed an algorithm that distributes tasks to the best computers in a way that gives sufficient performance to both cloud customers and providers in order to meet our objectives of lowering the total makespan of jobs on machines and providing improved quality of service. Based on specific criteria, the method is intended to select between the minmin scheduling algorithm and the maxmin algorithm implemented.

Pros:

- Less waiting time.
- local optimization
- Response time is minimized.

Cons:

- Not effective in load balancing
- Small makespan time.

1.2 Problem Statement

In order to minimise the execution time, or is MakeSpan time and resource cost the..total cost, the major challenge of task scheduling is to determine the optimal mapping of n independent tasks into m heterogenous resources. Maximising the resource utilization. Main goal of task schedul. creating map f to allocate (T), (R). i.e f:T->R.

- Tasks are independent
- The available resources (VMs) are heterogenous.

The project goal to min. the task execution time and execution cost as well increase resource utilization.

Main challenges are :

Resource Distribution Internet-based, cost-effective deployment of cloud resources to multiple applications.

The process of providing a service provider's resources to cloud users while assuring service quality is known as resource provisioning. This procedure is based on the service level agreement (SLA). It is classified as either Dynamic Resource Provisioning or Static Resource Provisioning.

Resource mapping: It is the alignment of resources needed by cloud customers with resources offered by a service provider.

Resource discovery and selection: The process of discovering all resources present in the system, gathering the current state of the resources, and deciding which target resources to select based on the information gathered from the discovery.

Resource adaption: is the capability of this system to dynamically adjust reources to meet the user requirements.

I. Enumeration Method

If all potential solutions are listed and weighed against one another, the best one for an optimization problem can be chosen. In the worst situation, exact enumerative algorithms have exponential time complexity. However, for a few weakly defined NP-hard problems, when the

When number of instance is small enough, it be solved using a pseudopolynomial method, the time complexity of which is constrained the input size and the problem's maximum number and their polynomial form. In addition, there is a different type of enumeration known as implicit enumeration, which examines every potential answer without explicitly naming them all. A useful implicit enumeration technique to handle combinational optimization issues is dynamic programming. It breaks down a problem into several stages, and The decisions that must be made at each stage have an effect on those that must be made at later stages. Since the number of stored decisions grows exponentially with the number of subproblems, exponential complexity is the worst type of complexity for dynamic programming algorithms.

II. Heuristic Method

Due to the fact that onlyfew particular situations of NP-hard problems have absolutely solution algorithms polynomial time, exhaustive enumeration is not practical for scheduling issues. We frequently settle for less-than-ideal solutions in order to balance accuracy and time since it's good practice.A heuristic is a suboptimal algorithm that locates moderately effective answers quickly. It does not ensure the optimal solution, but improves a candidate solution in relation to a specific criterion quality. In order to assess the precision of heuristic algorithms, approximation rate rH(e) is introduced .

III. Relaxation Method

Relaxing some of the original problem's constraints is another viable approach to solving NP-hard issues. The answer to the new relaxed problem might simple to find and have close approximation the solution to the problem. The typical easing includes:

• Assume that instead of real numbers, all the constituents in one instance are natural numbers.

• Assume that the value of one particular element doesn't change at all.

• Assume that two linked elements have equal values rather than that one of them is constrained by the other.

• Assume that an element's value is unit rather than being chosen at random.

• Assume that an element's type is predetermined rather than random.

1.3 Objective

1.3.1 Resource provisioning

When the people request for resources is approved by cloud service provider, a resource allocation technique is used to create and assign that user a practical number of virtual machines (VMs) based on demand. In addition, resource provisioning is in charge of matching incoming workloads or applications (cloudlets) to resources, user requirements based on QoS standards, and SLA agreements (VMs). In other words, it is important to provide incoming tasks and applications with the fewest resources possible while still ensuring a high level of quality of service.

1.3.2 Scheduling resources

The scheduling involves examining tnecessary QoS characteristics in order to decide which activity should be carried out.

- Selecting the most appropiate VM.
- ensuring the fulfillment of QoS.

The RPS handle the task scheduling , and aims to provision the VMs to users.

- understanding the constomer demands.

1.4 Methodology

There aren't many sources of canonical inputs or benchmarks to conduct comparison assessments because cloud computing is a relatively new sector. Running a benchmark additionally necessitates testing in a repeatable, dependable, and scalable environment, which is not feasible in the real world given the variety of cloud service providers and various planning policies. is attainable. and surroundings.

So, to get an overall software framework for modeling a cloud computing environment, and to run the tests, We employ a cloud simulator by the name of CloudSim. The flexible and expandable simulation framework that CloudSim offers makes it possible to model, simulate, and experiment with novel cloud computing infrastructure and application services with ease. With the aid of CloudSim, developers can concentrate on the specific system design issue wish to investigate without having to worry about the nitty-gritty specifics of cloud-based infrastructure and services.

We propose to use CloudSim to compare the Execution time . cost, makespan time to the other scheduling algorithms common inputs.

Algorithm Pseduo Code:-

To this end, use two registers to store the total burst time of all requests in the RQ, and use Areg to divide the value in SReg by the no. of task in the RQ. Stores the average burst time by column.

Inputs -SReg , AReg , Tn (task n), BT , TQ , Ready Queue

1.Begin

2. New requests Task arrives, tasks enter RQ

3. Update Sreg and Areg requests.

4. Task are loaded from RQ into VM queue be executed.

5.While (Ready Queue! = NULL) do

6. Ready queues T

7. Update Sreg&Areg

8. load T

9. if(readyQueue!=NULL) do

10. tq=BT(T)

11 incr. SReg& AReg

else

12. tq=avg(BT of all requests).

13. Sreg

14. // VM executes T by TQ Time

15. if(T terminated) then

16. Update Sreg & Areg

else Retrun T

18. Burst Time(BT)

19. Update AReg

20. end if

1.5 Organization

The content present in the prject report is designed into. five chapters.firstly the introduction chapter , chpater 2 describes literature survey, where different reseearch papers related to this project work have been included on task scheduling, resource allocation. Mostly explaned resource allocation , vm scheduling , task scheduling algorithms and considering the factors affecting task scheduling.

Chapter 3 summarizes the system design where various analytical and devlopment analysis is explained along with the design and algorithm of the mdel devepment. Model develoment technique is explained alnog with the design and agorithms of the model develpment. Model development technique has been explained using cloudsim simualtions.Web App framework development cycle.

Chapter 4 provides an account of the performance analysis where we have mentioned the most suitable algorithms for this prject after cmoparing the agrithms the basis of Makespantime, total execution cost, avg waiting time.

Chapter 5 reprents a brief summary of conclusion , and future work based on the research done during the implementation of the project.

In the end References is added where all the research papers are mentioned that were needed fot the better implementation of the algorithms.

Chapter 2-Literature survey

The best scheduling policy in the hydrid cloud model was proposed by R. Vanmechelen, and J. Broeckhove in 2010, according to a review of the literature. Both private and public clouds are included in the hybrid cloud. Private and public clouds make up hybrid. The situation of the issue is that, during periods of peak load, some workloads must be transferred from private cloud to public cloud since there aren't enough resources in the former to handle the tasks given by users. Both a deadline and QoS specifications place restrictions on these workloads. In this situation, a decision-making process is required to choose which workloads to which cloud provider, order to maximize amount of resources used in the internal data center and reduce operating costs.

In 2011, S. Sindhu and Saswati Mukherjee suggested algorithms for task scheduling in cloud computingon task processing requirements and resource computing power. The first algorithm, called (L CFPE), allocates longer cloudlets (tasks) than processing elements (PEs) with higher values and reduces the Makespan (time it takes to complete all tasks. total time). A second algorithm called Shortest Cloudlet Fastestb Processing

Dandhwani,Vanita,andVipul presented a K-means- task scheduling algorithm.The idea of the proposed algorithm is to use k-mean clustering technique to create clusters of tasks and assign clusters to VMs according to VM capacity. Results showed that the execution time and makespan of the were reduced, improving the overall system.

A allocation (PACO) method was proposed for work scheduling in the cloud system by Ruonan Lin and Li Qiang in 2016. The template size and the enhanced Ant Colony Optimization (ACO) method are used into this technique to individually plan activities. The simulated software tests the proposed algorithm and finds it to be rather efficient. The experiments show that the productivity of PACO Scheduling will improve the task.

2015, Moradbeiky A. Bardsiri conducted research using a task scheduling algorithm based on Cuckoo Optimization. In this algorithm, the bird's nest simulates processing unit (a virtual machine) and the egg is the task.

The parallel GA based approach for prioritizing the tasks when scheduling them in cloud systems was presented by Mehran Ashouraei and colleagues in 2018 (Mehran Ashouraei, 2018). This approach aims to use resources efficiently and reduce resource waste. In order to eliminate work failures, this method involves increasing load balance choosing qualified resources for urgent task.

Element (SCFP) does the opposite. SCFP maps short cloudlet process. Using this algorithm minimizes lead times and avoids running out of longer orders. They suggested that future work should use heuristics to test more algorithms and consider task priorities. It is used to optimize scheduling under the cloud . ACO resembles the behavior of ant colonies, in which ants move in random directions in search of food sources. In task scheduling, tasks such as ants and virtual machines mimic food sources.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 Software Specifications

All algorithms were written in Java 8 as .java files.Java was chosen because it is a powerful compiler. Java supports both Oops execution and debugging at the same time.Furthermore, Java is platform independent, unlike other languages like C++, C, which compile onto platform but Java is write-once, run-anywhere language. Java differs from other platform it is software based on the other platforms based software, ithas two components.

Java programming is platform-independent. Linux, Mac OS, Sun Solaris, Windows, etc. A compiler compiles Java code and turns it into bytecode. This bytecode can run on various platforms, making it platform independent code. Run anywhere and write once.

The general purpose, interactive, object-oriented, and high-level programming language python is particularly liked by global users, Python is dynamically and garbage collected.

Docker system software letting users to deploy,build,testing the web applications,Software is packaged by docker into standardised units called containers that contain all of the necessary libraries, system tools, code, and runtime for the software to function.

3.1.1 Cloudsim Toolkit

Cloudism is open source JDK tool originally distributed ,laboratory in the melbourne university. Cloudsim was used to simul the components of a cloud computing system, evaluate resource provisioning policies, test various Cloud Computing deployments, and generate a mix of workload requirements distribution. , to measure the performance of CC configure. and test scenario. These capabilities provided by CloudSim, Cloud sim developers to effiectively address and operate several key issues related to cloud computing and ultimately develops best results.

The complex real-world CC environment is modelled using the Cloudsim framework, which also simulates its behavior. With the help of this tool, a model is made up of many elements, including datacenters, hosts, service brokers, scheduling, and allocation policies. Cloud computing will develop.

To start using your CC system, you must have at least one data center listed in the Cloud Information Service Registry (CIS). Each data center creates a number of hosts. After receiving the list of cloudlets and virtual machines that have been submitted, the Data Center Broker applies allocation policy assign servers to VMs.

The CloudSim software is impl. using a layer design as shown in below Figure3.3

The first layer is simjava, which provides core functionalityto the top cloud layers, such as creating cloud systems and communicating between them. increase, Components simulation clock management, andqueuing and event handling.

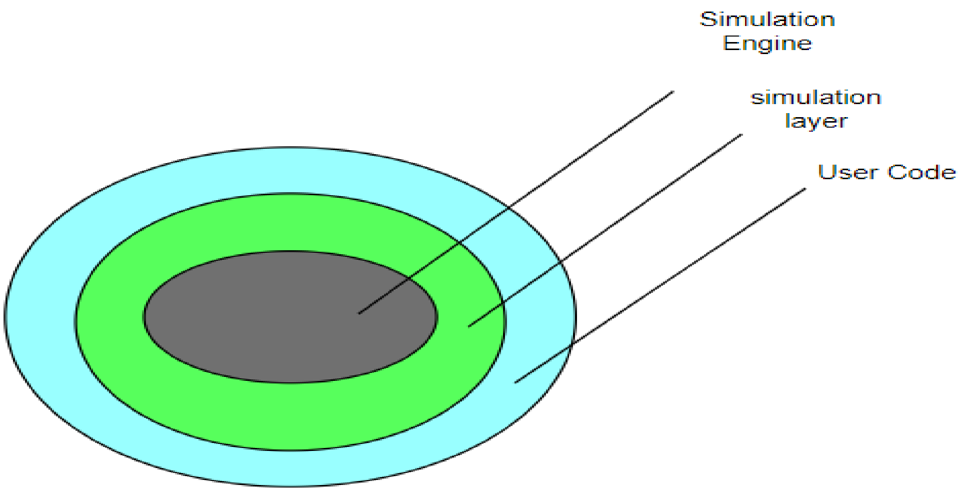


Fig 3.3 CloudSim layers

Eclipse

An integrated development environment called Eclipse is employed in computer programming. The environment can be customized using an extendable plug-in system and a base workspace. It is the second-most widely used IDE for developing Java.

3.2 Hardware Specification

Algorithms that are platform dependent have been implemented. They may be run on any of the Windows,Mac systems. We wrote the coding for our initial prototype in Eclipse. The GPU was utilized to build the model.

3.3 Input

To ensure the relevance of a simulation-based evaluation, investigations must be conducted utilizing workload are from real world . Input data for CPU utl. for over 100 VMs from servers .20 tasks, 20 cludlets The utilization measurement interval is 5 minutes, each tracked file has, so represents approximately 24,44 hours of VM CPU utilization.

For Task scheduling simulator the input are given by user like:

- execution time
- arrival time
- period
- deadline

3.4 Output

It is noted that SLA violations and energy usage occur for activities that range in size from sets of 10 to 100.To determine the true statistical performance difference between the methods, simulation is performed and result are drawn that compare the results from the suggested method and existing scheduling algorithms.the make span time is considered in the performance and comparison of schedling algorithms.

The simulator gives output result in the form of graph.

3.5 Algorithm Designs

Tasks T = to, t1.. t2.. tN-1, and no.of tasks is N=|T| whereas the no of hosts is m=|H| for the set of hosts H = to, h0, h1.. hn-1, The avg. load of the VM that run a host system is referred to as the host machine's load.

Step 1: Set the host resource H={h0,h1…hm-1} and sort in asc. order of the processing power.

Step 2: Considering the VM, choose the host resource (that can handle the least amount of load while providing the necessary resources. If the host is already present, create the virtual machine and allocate the appropriate resources for it, then update the resources that are now accessible. Move Task t1 to the end of the task queue in the absence of a host and wait for the next scheduling.

Step 3: If the Task t i's resource requirements rise, determine whether the host that Task t i's virtual machine runs on can accommodate the extra resources needed, if so, allocate those resources to the host, modify the virtual machine's configuration before updating the host's resource availability.
If not, the virtual machine is transferred to the host that experiences the least load and has the additional resources required to run continuously.

Step 4: If resourcess requires task t1 reduce release the overlaod resources so VM occupied and update the host.

Step 5: If taks t1 has been completed then destroys then destroy the virtual machine of task t1 realse the occupied resource for the other unfinished tasks.

Step 6: Find the SDV. of each host's load. Select one virtual serer from a host and relocate it to host with the lowest load if one host has a significantly higher load than the others. If a host is under very low stress, move all virtual machines (VMs) from that host to another host.

Step 7:

Determine lad on every host and its std.deviation. If one host has a much larger load than the others, choose a virtual machine from that host and move it to the host with the lowest load. Move all virtual machines (VMs) from a given host if it is experiencing very little load to another host.

3.6 Creating Web Application

With the increase of technical aspect of life,we aim to create an scheduling simulator for task scheduling,
with one platform for scheduling simulation for all scheduling algorithms, for giving an idea of how simulation works.

So to make our application user ready we have created an interface using technologies like HTML,
CSS,Django.

3.6.1 HTML

HTML stands for Hyper Text Markup Language. It will act as the skeleton for Web page and its design.

3.6.2 CSS

It stands for Cascading Style Sheet. It is mainly used to beautify our web page.

3.6.3 Django

A Python-based web framework called Django enables you to easily build effective online apps. Django has built-in functionality for everything, including the Django Admin Interface and the default SQLlite3 database, it is often known as the batteries included framework.It is highly scalable,easy to develop web app and learn.

Flowchart

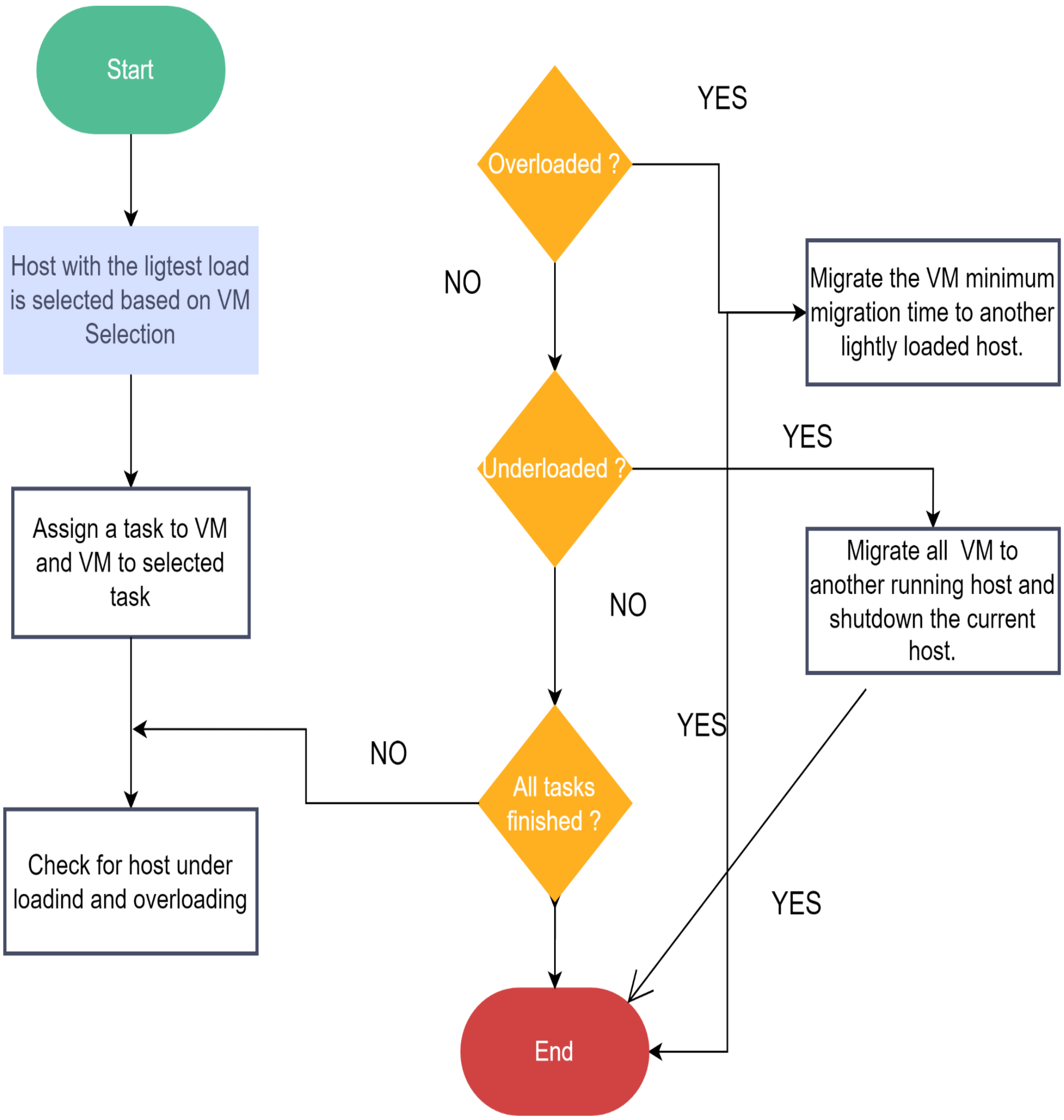


Fig 3.4 System Design Flowchart

Chapter 4: Performance and Analysis

4.1 Algortihms Results and Analysis

1.FCFS Algortihms:

```
<terminated> FCFS_Scheduler [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v2022051
Datacenter_18 is shutting down...
Datacenter_19 is shutting down...
Broker_0 is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time    WaitingTime    CompletionTime    Cost
    38    SUCCESS    03    03    00    00.1    00.1    00    00    00
    06    SUCCESS    07    07    00    00.1    00.1    00    00    00
    10    SUCCESS    08    08    00    00.1    00.1    00    00    00
    02    SUCCESS    09    09    00    00.1    00.1    00    00    00
    07    SUCCESS    10    10    00    00.1    00.1    00    00    00
    15    SUCCESS    11    11    00    00.1    00.1    00    00    00
    11    SUCCESS    12    12    00    00.1    00.1    00    00    00
    08    SUCCESS    13    13    00    00.1    00.1    00    00    00
    04    SUCCESS    14    14    00    00.1    00.1    00    00    00
    05    SUCCESS    15    15    00    00.1    00.1    00    00    00
    12    SUCCESS    16    16    00    00.1    00.1    00    00    00
    42    SUCCESS    17    17    00    00.1    00.1    00    00    00
    28    SUCCESS    18    18    00    00.1    00.1    00    00    00
    00    SUCCESS    19    19    00    00.1    00.1    00    00    00
    14    SUCCESS    20    20    00    00.1    00.1    00    00    00
    41    SUCCESS    21    21    00    00.1    00.1    00    00    00
    39    SUCCESS    03    03    00.11    00.1    00.21    00    00.11    00.33
    36    SUCCESS    07    07    00.11    00.1    00.21    00    00.11    00.33
    20    SUCCESS    08    08    00.11    00.1    00.21    00    00.11    00.33
    32    SUCCESS    09    09    00.11    00.1    00.21    00    00.11    00.33
    17    SUCCESS    10    10    00.11    00.1    00.21    00    00.11    00.33
    23    SUCCESS    11    11    00.11    00.1    00.21    00    00.11    00.33
    18    SUCCESS    12    12    00.11    00.1    00.21    00    00.11    00.33
    25    SUCCESS    13    13    00.11    00.1    00.21    00    00.11    00.33
    44    SUCCESS    15    15    00.11    00.1    00.21    00    00.11    00.33
    30    SUCCESS    16    16    00.11    00.1    00.21    00    00.11    00.33
    01    SUCCESS    19    19    00.11    00.1    00.21    00    00.11    00.33
    16    SUCCESS    20    20    00.11    00.1    00.21    00    00.11    00.33
    48    SUCCESS    09    09    00.11    00.21    00.32    00.11    00.22    00.33
    22    SUCCESS    10    10    00.11    00.21    00.32    00.11    00.22    00.33
    47    SUCCESS    11    11    00.11    00.21    00.32    00.11    00.22    00.33
    26    SUCCESS    12    12    00.11    00.21    00.32    00.11    00.22    00.33
```

```
<terminated> FCFS_Scheduler [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\
    39   SUCCESS   03   03   00.11    00.1      00.21     00       00.11    00.33
    36   SUCCESS   07   07   00.11    00.1      00.21     00       00.11    00.33
    20   SUCCESS   08   08   00.11    00.1      00.21     00       00.11    00.33
    32   SUCCESS   09   09   00.11    00.1      00.21     00       00.11    00.33
    17   SUCCESS   10   10   00.11    00.1      00.21     00       00.11    00.33
    23   SUCCESS   11   11   00.11    00.1      00.21     00       00.11    00.33
    18   SUCCESS   12   12   00.11    00.1      00.21     00       00.11    00.33
    25   SUCCESS   13   13   00.11    00.1      00.21     00       00.11    00.33
    44   SUCCESS   15   15   00.11    00.1      00.21     00       00.11    00.33
    30   SUCCESS   16   16   00.11    00.1      00.21     00       00.11    00.33
    01   SUCCESS   19   19   00.11    00.1      00.21     00       00.11    00.33
    16   SUCCESS   20   20   00.11    00.1      00.21     00       00.11    00.33
    48   SUCCESS   09   09   00.11    00.21     00.32     00.11    00.22    00.33
    22   SUCCESS   10   10   00.11    00.21     00.32     00.11    00.22    00.33
    47   SUCCESS   11   11   00.11    00.21     00.32     00.11    00.22    00.33
    26   SUCCESS   12   12   00.11    00.21     00.32     00.11    00.22    00.33
    27   SUCCESS   13   13   00.11    00.21     00.32     00.11    00.22    00.33
    31   SUCCESS   16   16   00.11    00.21     00.32     00.11    00.22    00.33
    03   SUCCESS   19   19   00.11    00.21     00.32     00.11    00.22    00.33
    24   SUCCESS   20   20   00.11    00.21     00.32     00.11    00.22    00.33
    29   SUCCESS   12   12   00.11    00.32     00.43     00.22    00.33    00.33
    35   SUCCESS   13   13   00.11    00.32     00.43     00.22    00.33    00.33
    34   SUCCESS   16   16   00.11    00.32     00.43     00.22    00.33    00.33
    46   SUCCESS   19   19   00.11    00.32     00.43     00.22    00.33    00.33
    40   SUCCESS   20   20   00.11    00.32     00.43     00.22    00.33    00.33
    33   SUCCESS   12   12   00.11    00.43     00.54     00.33    00.44    00.33
    43   SUCCESS   13   13   00.11    00.43     00.54     00.33    00.44    00.33
    49   SUCCESS   16   16   00.11    00.43     00.54     00.33    00.44    00.33
    45   SUCCESS   13   13   00.11    00.54     00.65     00.44    00.55    00.33
    09   SUCCESS   04   04   522.55   00.1      522.65    00       522.55   1567.64
    21   SUCCESS   02   02   637.97   00.1      638.07    00       637.97   1913.9
    37   SUCCESS   02   02   00.11    638.07    638.18    637.97   638.08   00.33
    13   SUCCESS   04   04   715.46   522.65    1238.11   522.55   1238.01  2146.38
    19   SUCCESS   04   04   666.37   1238.11   1904.47   1238.01  1904.38  1999.1
Makespan using FCFS: 1221.2232289234394
Total Completion Time: 4947.59 Avg Completion Time: 98.9518
Total Cost : 7636.940999999999 Avg cost: 152.73881999999998
Avg Waiting Time: 48.03886000000001
org.cloudbus.cloudsim.examples.FCFS_Scheduler finished!
```

Makespan using FCFS: 1221.2232289234394

Total Completion Time: 4947.59 Avg Completion Time: 98.9518

Total Cost : 7636.940999999999 Avg cost: 152.73881999999998

Avg Waiting Time: 48.0388600000000

2.MinMin algorithm

Simulation completed.

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
48	SUCCESS	5	5	0	0.1	0.1
0	SUCCESS	7	7	0	0.1	0.1
24	SUCCESS	8	8	0	0.1	0.1
23	SUCCESS	9	9	0	0.1	0.1
10	SUCCESS	10	10	0	0.1	0.1
27	SUCCESS	11	11	0	0.1	0.1
8	SUCCESS	12	12	0	0.1	0.1
12	SUCCESS	13	13	0	0.1	0.1
15	SUCCESS	14	14	0	0.1	0.1
3	SUCCESS	15	15	0	0.1	0.1
29	SUCCESS	16	16	0	0.1	0.1
21	SUCCESS	17	17	0	0.1	0.1
30	SUCCESS	18	18	0	0.1	0.1
1	SUCCESS	19	19	0	0.1	0.1
20	SUCCESS	20	20	0	0.1	0.1
2	SUCCESS	21	21	0	0.1	0.1
19	SUCCESS	7	7	0.11	0.1	0.21
13	SUCCESS	10	10	0.11	0.1	0.21
36	SUCCESS	11	11	0.11	0.1	0.21
25	SUCCESS	12	12	0.11	0.1	0.21
28	SUCCESS	13	13	0.11	0.1	0.21
41	SUCCESS	14	14	0.11	0.1	0.21
31	SUCCESS	16	16	0.11	0.1	0.21
43	SUCCESS	17	17	0.11	0.1	0.21
11	SUCCESS	19	19	0.11	0.1	0.21
7	SUCCESS	21	21	0.11	0.1	0.21
32	SUCCESS	7	7	0.11	0.21	0.32
17	SUCCESS	10	10	0.11	0.21	0.32
46	SUCCESS	11	11	0.11	0.21	0.32
39	SUCCESS	12	12	0.11	0.21	0.32
35	SUCCESS	13	13	0.11	0.21	0.32
14	SUCCESS	19	19	0.11	0.21	0.32
34	SUCCESS	10	10	0.11	0.32	0.43
45	SUCCESS	12	12	0.11	0.32	0.43
22	SUCCESS	19	19	0.11	0.32	0.43

13	SUCCESS	10	10	0.11	0.1	0.21
36	SUCCESS	11	11	0.11	0.1	0.21
25	SUCCESS	12	12	0.11	0.1	0.21
28	SUCCESS	13	13	0.11	0.1	0.21
41	SUCCESS	14	14	0.11	0.1	0.21
31	SUCCESS	16	16	0.11	0.1	0.21
43	SUCCESS	17	17	0.11	0.1	0.21
11	SUCCESS	19	19	0.11	0.1	0.21
7	SUCCESS	21	21	0.11	0.1	0.21
32	SUCCESS	7	7	0.11	0.21	0.32
17	SUCCESS	10	10	0.11	0.21	0.32
46	SUCCESS	11	11	0.11	0.21	0.32
39	SUCCESS	12	12	0.11	0.21	0.32
35	SUCCESS	13	13	0.11	0.21	0.32
14	SUCCESS	19	19	0.11	0.21	0.32
34	SUCCESS	10	10	0.11	0.32	0.43
45	SUCCESS	12	12	0.11	0.32	0.43
22	SUCCESS	19	19	0.11	0.32	0.43
37	SUCCESS	10	10	0.11	0.43	0.54
47	SUCCESS	12	12	0.11	0.43	0.54
42	SUCCESS	19	19	0.11	0.43	0.54
49	SUCCESS	10	10	0.11	0.54	0.65
4	SUCCESS	4	4	122.08	0.1	122.18
33	SUCCESS	4	4	0.11	122.18	122.29
40	SUCCESS	4	4	0.11	122.29	122.4
44	SUCCESS	4	4	0.11	122.4	122.51
9	SUCCESS	6	6	316.06	0.1	316.16
6	SUCCESS	3	3	605.83	0.1	605.93
5	SUCCESS	2	2	779.19	0.1	779.29
18	SUCCESS	3	3	278.9	605.93	884.83
16	SUCCESS	6	6	742.39	316.16	1058.56
26	SUCCESS	3	3	943.01	884.83	1827.84
38	SUCCESS	3	3	0.11	1827.84	1827.95

```
Makespan using Minmin: 1208.4114221377815
Total Completion Time: 7794.141 Avg Completion Time: 155.88281999999998
Total Cost : 11371.329000000002 Avg cost: 227.42658000000003
Avg Waiting Time: 80.07396
org.cloudbus.cloudsim.examples.MinMin finished!
```

Makespan using Minmin: 1208.4114221377815

Total Completion Time: 7794.141 Avg Completion Time: 155.88281999999998

Total Cost : 11371.329000000002 Avg cost: 227.42658000000003

Avg Waiting Time: 80.07396

3.MaxMin Algorithm

Simulation completed.

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
45	SUCCESS	4	4	0	0.1	0.1
35	SUCCESS	5	5	0	0.1	0.1
12	SUCCESS	7	7	0	0.1	0.1
2	SUCCESS	8	8	0	0.1	0.1
4	SUCCESS	9	9	0	0.1	0.1
1	SUCCESS	10	10	0	0.1	0.1
17	SUCCESS	11	11	0	0.1	0.1
9	SUCCESS	12	12	0	0.1	0.1
19	SUCCESS	13	13	0	0.1	0.1
11	SUCCESS	14	14	0	0.1	0.1
7	SUCCESS	15	15	0	0.1	0.1
10	SUCCESS	16	16	0	0.1	0.1
8	SUCCESS	17	17	0	0.1	0.1
28	SUCCESS	18	18	0	0.1	0.1
14	SUCCESS	19	19	0	0.1	0.1
0	SUCCESS	20	20	0	0.1	0.1
15	SUCCESS	21	21	0	0.1	0.1
31	SUCCESS	7	7	0.11	0.1	0.21
26	SUCCESS	8	8	0.11	0.1	0.21
16	SUCCESS	10	10	0.11	0.1	0.21
22	SUCCESS	11	11	0.11	0.1	0.21
21	SUCCESS	12	12	0.11	0.1	0.21
29	SUCCESS	13	13	0.11	0.1	0.21
30	SUCCESS	14	14	0.11	0.1	0.21
24	SUCCESS	16	16	0.11	0.1	0.21
23	SUCCESS	19	19	0.11	0.1	0.21
6	SUCCESS	20	20	0.11	0.1	0.21
37	SUCCESS	10	10	0.11	0.21	0.32
33	SUCCESS	11	11	0.11	0.21	0.32
25	SUCCESS	12	12	0.11	0.21	0.32
34	SUCCESS	13	13	0.11	0.21	0.32
49	SUCCESS	16	16	0.11	0.21	0.32
39	SUCCESS	19	19	0.11	0.21	0.32
13	SUCCESS	20	20	0.11	0.21	0.32
38	SUCCESS	10	10	0.11	0.32	0.43

```
<terminated> MaxMin [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\
    31        SUCCESS        7            7         0.11        0.1        0.21
    26        SUCCESS        8            8         0.11        0.1        0.21
    16        SUCCESS        10           10          0.11        0.1        0.21
    22        SUCCESS        11           11          0.11        0.1        0.21
    21        SUCCESS        12           12          0.11        0.1        0.21
    29        SUCCESS        13           13          0.11        0.1        0.21
    30        SUCCESS        14           14          0.11        0.1        0.21
    24        SUCCESS        16           16          0.11        0.1        0.21
    23        SUCCESS        19           19          0.11        0.1        0.21
    6        SUCCESS        20           20         0.11        0.1        0.21
    37        SUCCESS        10           10          0.11        0.21        0.32
    33        SUCCESS        11           11          0.11        0.21        0.32
    25        SUCCESS        12           12          0.11        0.21        0.32
    34        SUCCESS        13           13          0.11        0.21        0.32
    49        SUCCESS        16           16          0.11        0.21        0.32
    39        SUCCESS        19           19          0.11        0.21        0.32
    13        SUCCESS        20           20          0.11        0.21        0.32
    38        SUCCESS        10           10          0.11        0.32        0.43
    32        SUCCESS        12           12          0.11        0.32        0.43
    44        SUCCESS        20           20          0.11        0.32        0.43
    42        SUCCESS        10           10          0.11        0.43        0.54
    36        SUCCESS        12           12          0.11        0.43        0.54
    46        SUCCESS        20           20          0.11        0.43        0.54
    48        SUCCESS        20           20          0.11        0.54        0.65
    27        SUCCESS        6            6         184.73        0.1        184.83
    41        SUCCESS        6            6         0.11        184.83        184.94
    18        SUCCESS        3            3         278.9        0.1        279
    40        SUCCESS        3            3         0.11        279        279.11
    43        SUCCESS        3            3         0.11        279.11        279.22
    3        SUCCESS        2            2         663.04        0.1        663.14
    5        SUCCESS        2            2         779.19        663.14        1442.33
    20        SUCCESS        2            2         388.4        1442.33        1830.73
    47        SUCCESS        2            2         0.11        1830.73        1830.84
Makespan using Maxmin: 1354.5280235668106
Total Completion Time: 6978.762000000001 Avg Completion Time: 139.57524
Total Cost : 6892.053000000001 Avg cost: 137.84106000000003
Avg Waiting Time: 93.62822
org.cloudbus.cloudsim.examples.Maxmin finished!
```

Makespan using Maxmin: 1354.5280235668106

Total Completion Time: 6978.762000000001 Avg Completion Time: 139.57524

Total Cost : 6892.053000000001 Avg cost: 137.84106000000003

Avg Waiting Time: 93.62822

4. Round Robin Algorithm

<terminated> RoundRobinScheduler [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220

Simulation completed.

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	WaitingTime	CompletionTime	Cost
06	SUCCESS	07	07	00	00.1	00.1	00	00	00
28	SUCCESS	08	08	00	00.1	00.1	00	00	00
10	SUCCESS	09	09	00	00.1	00.1	00	00	00
04	SUCCESS	10	10	00	00.1	00.1	00	00	00
13	SUCCESS	11	11	00	00.1	00.1	00	00	00
00	SUCCESS	12	12	00	00.1	00.1	00	00	00
46	SUCCESS	13	13	00	00.1	00.1	00	00	00
02	SUCCESS	14	14	00	00.1	00.1	00	00	00
03	SUCCESS	15	15	00	00.1	00.1	00	00	00
31	SUCCESS	16	16	00	00.1	00.1	00	00	00
09	SUCCESS	17	17	00	00.1	00.1	00	00	00
38	SUCCESS	18	18	00	00.1	00.1	00	00	00
01	SUCCESS	19	19	00	00.1	00.1	00	00	00
18	SUCCESS	20	20	00	00.1	00.1	00	00	00
15	SUCCESS	21	21	00	00.1	00.1	00	00	00
25	SUCCESS	07	07	00.11	00.1	00.21	00	00.11	00.33
21	SUCCESS	09	09	00.11	00.1	00.21	00	00.11	00.33
33	SUCCESS	11	11	00.11	00.1	00.21	00	00.11	00.33
24	SUCCESS	12	12	00.11	00.1	00.21	00	00.11	00.33
19	SUCCESS	14	14	00.11	00.1	00.21	00	00.11	00.33
08	SUCCESS	15	15	00.11	00.1	00.21	00	00.11	00.33
42	SUCCESS	16	16	00.11	00.1	00.21	00	00.11	00.33
11	SUCCESS	17	17	00.11	00.1	00.21	00	00.11	00.33
20	SUCCESS	19	19	00.11	00.1	00.21	00	00.11	00.33
27	SUCCESS	20	20	00.11	00.1	00.21	00	00.11	00.33
16	SUCCESS	21	21	00.11	00.1	00.21	00	00.11	00.33
40	SUCCESS	07	07	00.11	00.21	00.32	00.11	00.22	00.33
23	SUCCESS	09	09	00.11	00.21	00.32	00.11	00.22	00.33
47	SUCCESS	14	14	00.11	00.21	00.32	00.11	00.22	00.33
30	SUCCESS	15	15	00.11	00.21	00.32	00.11	00.22	00.33
45	SUCCESS	17	17	00.11	00.21	00.32	00.11	00.22	00.33
26	SUCCESS	19	19	00.11	00.21	00.32	00.11	00.22	00.33
39	SUCCESS	20	20	00.11	00.21	00.32	00.11	00.22	00.33
22	SUCCESS	21	21	00.11	00.21	00.32	00.11	00.22	00.33
35	SUCCESS	09	09	00.11	00.32	00.43	00.22	00.33	00.33

```
<terminated> RoundRobinScheduler [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v2022051
    33      SUCCESS     11      11      00.11     00.1       00.21      00       00.11     00.33
    24      SUCCESS     12      12      00.11     00.1       00.21      00       00.11     00.33
    19      SUCCESS     14      14      00.11     00.1       00.21      00       00.11     00.33
    08      SUCCESS     15      15      00.11     00.1       00.21      00       00.11     00.33
    42      SUCCESS     16      16      00.11     00.1       00.21      00       00.11     00.33
    11      SUCCESS     17      17      00.11     00.1       00.21      00       00.11     00.33
    20      SUCCESS     19      19      00.11     00.1       00.21      00       00.11     00.33
    27      SUCCESS     20      20      00.11     00.1       00.21      00       00.11     00.33
    16      SUCCESS     21      21      00.11     00.1       00.21      00       00.11     00.33
    40      SUCCESS     07      07      00.11     00.21      00.32      00.11    00.22     00.33
    23      SUCCESS     09      09      00.11     00.21      00.32      00.11    00.22     00.33
    47      SUCCESS     14      14      00.11     00.21      00.32      00.11    00.22     00.33
    30      SUCCESS     15      15      00.11     00.21      00.32      00.11    00.22     00.33
    45      SUCCESS     17      17      00.11     00.21      00.32      00.11    00.22     00.33
    26      SUCCESS     19      19      00.11     00.21      00.32      00.11    00.22     00.33
    39      SUCCESS     20      20      00.11     00.21      00.32      00.11    00.22     00.33
    22      SUCCESS     21      21      00.11     00.21      00.32      00.11    00.22     00.33
    35      SUCCESS     09      09      00.11     00.32      00.43      00.22    00.33     00.33
    49      SUCCESS     14      14      00.11     00.32      00.43      00.22    00.33     00.33
    32      SUCCESS     19      19      00.11     00.32      00.43      00.22    00.33     00.33
    36      SUCCESS     09      09      00.11     00.43      00.54      00.33    00.44     00.33
    34      SUCCESS     19      19      00.11     00.43      00.54      00.33    00.44     00.33
    43      SUCCESS     09      09      00.11     00.54      00.65      00.44    00.55     00.33
    48      SUCCESS     19      19      00.11     00.54      00.65      00.44    00.55     00.33
    12      SUCCESS     05      05      184.97    00.1       185.07     00       184.97    554.91
    41      SUCCESS     05      05      00.11     185.07     185.18     184.97   185.08    00.33
    05      SUCCESS     06      06      225.89    00.1       225.99     00       225.89    677.66
    44      SUCCESS     06      06      00.11     225.99     226.1      225.89   226       00.33
    17      SUCCESS     03      03      479.87    00.1       479.97     00       479.87    1439.6
    07      SUCCESS     02      02      513.88    00.1       513.98     00       513.88    1541.64
    29      SUCCESS     02      02      322.75    513.98     836.73     513.88   836.63    968.24
    14      SUCCESS     04      04      1005.4    00.1       1005.5     00       1005.4    3016.19
    37      SUCCESS     04      04      00.11     1005.5     1005.61    1005.4   1005.51   00.33
Makespan using RR: 1311.9290535570512
Total Completion Time: 4669.169 Avg Completion Time: 93.38338
Total Cost : 8207.826000000001 Avg cost: 164.15652000000003
Avg Waiting Time: 38.664539999999995
org.cloudbus.cloudsim.examples.RoundRobinScheduler finished!
```

Makespan using RR: 1311.9290535570512

Total Completion Time: 4669.169 Avg Completion Time: 93.38338

Total Cost : 8207.826000000001 Avg cost: 164.15652000000003

Avg Waiting Time: 38.664539999999995

5.Shortest Job First :

```
<terminated> SJF_Scheduler (2) [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.c
    27        SUCCESS      11        11        00.11      00.1        00.21
    17        SUCCESS      13        13        00.11      00.1        00.21
    37        SUCCESS      15        15        00.11      00.1        00.21
    32        SUCCESS      16        16        00.11      00.1        00.21
    33        SUCCESS      17        17        00.11      00.1        00.21
    23        SUCCESS      19        19        00.11      00.1        00.21
    28        SUCCESS      20        20        00.11      00.1        00.21
    46        SUCCESS      05        05        00.11      00.21        00.32
    44        SUCCESS      07        07        00.11      00.21        00.32
    12        SUCCESS      08        08        00.11      00.21        00.32
    08        SUCCESS      10        10        00.11      00.21        00.32
    34        SUCCESS      11        11        00.11      00.21        00.32
    24        SUCCESS      13        13        00.11      00.21        00.32
    38        SUCCESS      15        15        00.11      00.21        00.32
    43        SUCCESS      16        16        00.11      00.21        00.32
    42        SUCCESS      17        17        00.11      00.21        00.32
    36        SUCCESS      19        19        00.11      00.21        00.32
    49        SUCCESS      20        20        00.11      00.21        00.32
    21        SUCCESS      08        08        00.11      00.32        00.43
    13        SUCCESS      10        10        00.11      00.32        00.43
    47        SUCCESS      15        15        00.11      00.32        00.43
    48        SUCCESS      16        16        00.11      00.32        00.43
    45        SUCCESS      19        19        00.11      00.32        00.43
    18        SUCCESS      10        10        00.11      00.43        00.54
    41        SUCCESS      10        10        00.11      00.54        00.65
    00        SUCCESS      06        06        1169.54     00.1        1169.64
    11        SUCCESS      06        06        1095.08     1169.64       2264.72
    02        SUCCESS      03        03        2920.54     00.1        2920.64
    04        SUCCESS      03        03        2306.94     2920.64       5227.58
    15        SUCCESS      03        03        1783.5     5227.58       7011.08
    31        SUCCESS      03        03        00.11     7011.08       7011.19
    39        SUCCESS      03        03        00.11     7011.19       7011.3
Makespan using SJF: 1490.0679364570929
Total Completion Time: 32621.78 Avg Completion Time: 652.4356
Total Cost : 27837.066000000006 Avg cost: 556.7413200000001
Avg Waiting Time: 466.85515999999996
org.cloudbus.cloudsim.examples.SJF_Scheduler finished!
```

```
<terminated> SJF_Scheduler (2) [Java Application] C:\Users\Shubham\Desktop\cloud simsetup\eclipse-java-2022-06-R-win32-x86_64\eclipse\plugin
Datacenter_16 is shutting down...
Datacenter_17 is shutting down...
Datacenter_18 is shutting down...
Datacenter_19 is shutting down...
Broker_0 is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS     Data center ID    VM ID       Time     Start Time     Finish Time
    35        SUCCESS        05              05        00        00.1          00.1
    03        SUCCESS        07              07        00        00.1          00.1
    01        SUCCESS        08              08        00        00.1          00.1
    05        SUCCESS        10              10        00        00.1          00.1
    14        SUCCESS        11              11        00        00.1          00.1
    19        SUCCESS        12              12        00        00.1          00.1
    16        SUCCESS        13              13        00        00.1          00.1
    29        SUCCESS        15              15        00        00.1          00.1
    20        SUCCESS        16              16        00        00.1          00.1
    30        SUCCESS        17              17        00        00.1          00.1
    25        SUCCESS        18              18        00        00.1          00.1
    22        SUCCESS        19              19        00        00.1          00.1
    09        SUCCESS        20              20        00        00.1          00.1
    26        SUCCESS        21              21        00        00.1          00.1
    40        SUCCESS        05              05        00.11        00.1          00.21
    07        SUCCESS        07              07        00.11        00.1          00.21
    10        SUCCESS        08              08        00.11        00.1          00.21
    06        SUCCESS        10              10        00.11        00.1          00.21
    27        SUCCESS        11              11        00.11        00.1          00.21
    17        SUCCESS        13              13        00.11        00.1          00.21
    37        SUCCESS        15              15        00.11        00.1          00.21
    32        SUCCESS        16              16        00.11        00.1          00.21
    33        SUCCESS        17              17        00.11        00.1          00.21
    23        SUCCESS        19              19        00.11        00.1          00.21
    28        SUCCESS        20              20        00.11        00.1          00.21
    46        SUCCESS        05              05        00.11        00.21          00.32
    44        SUCCESS        07              07        00.11        00.21          00.32
    12        SUCCESS        08              08        00.11        00.21          00.32
    08        SUCCESS        10              10        00.11        00.21          00.32
```

Makespan using SJF: 1490.0679364570929

Total Completion Time: 32621.78 Avg Completion Time: 652.4356

Total Cost : 27837.066000000006 Avg cost: 556.7413200000001

Avg Waiting Time: 466.85515999999996

6. Selective Algortihm

```
0.1: Broker: Sending cloudlet 8 to VM #3
0.1: Broker: Sending cloudlet 9 to VM #3
44.544444444444444: Broker: Cloudlet 3 received
54.526666666666664: Broker: Cloudlet 6 received
63.41555555555556: Broker: Cloudlet 4 received
71.19333333333333: Broker: Cloudlet 7 received
77.86: Broker: Cloudlet 5 received
83.41555555555556: Broker: Cloudlet 9 received
87.86: Broker: Cloudlet 0 received
91.19333333333333: Broker: Cloudlet 8 received
93.41555555555556: Broker: Cloudlet 1 received
95.63777777777779: Broker: Cloudlet 2 received
95.63777777777779: Broker: All Cloudlets executed. Finishing...
95.63777777777779: Broker: Destroying VM #0
95.63777777777779: Broker: Destroying VM #1
95.63777777777779: Broker: Destroying VM #2
95.63777777777779: Broker: Destroying VM #3
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID     STATUS     Data center ID    VM ID     Time     Start Time     Finish Time     Algo
    3          SUCCESS          2              3        44.44       0.1          44.54          Max-min
    6          SUCCESS          2              3        54.43       0.1          54.53          Max-min
    4          SUCCESS          2              3        63.32       0.1          63.42          Max-min
    7          SUCCESS          2              3        71.09       0.1          71.19          Max-min
    5          SUCCESS          2              3        77.76       0.1          77.86          Min-min
    9          SUCCESS          2              3        83.32       0.1          83.42          Min-min
    0          SUCCESS          2              3        87.76       0.1          87.86          Max-min
    8          SUCCESS          2              3        91.09       0.1          91.19          Max-min
    1          SUCCESS          2              3        93.32       0.1          93.42          Max-min
    2          SUCCESS          2              3        95.54       0.1          95.64          Max-min
Selective Algorithm finished!
```

4.2 Result and Comparison:

There is the comparison between Various Task scheduling algorithms.

Algortihms	MakeSpan Time	AvgCompletion Time	Avg Waiting time
SJF	1490.06	652.43	466.85
FCFS	1221.22	98.95	48.03
RR	1311.92	93.38	38.66
MaxMin	1354.52	139.57	93.62
MinMin	1208.41	155.88	80.07
Selective	1280.54	140.45	81.34

The results above show that considering MakeSpan time MinMin Algorithms is efficient followed by FCFS algorithms.

- Taking Avg completion RR algorithms shows the best result followed by FCFS.
- Taking account Avg Waiting time RR algorithm is the best followed by FCFS.

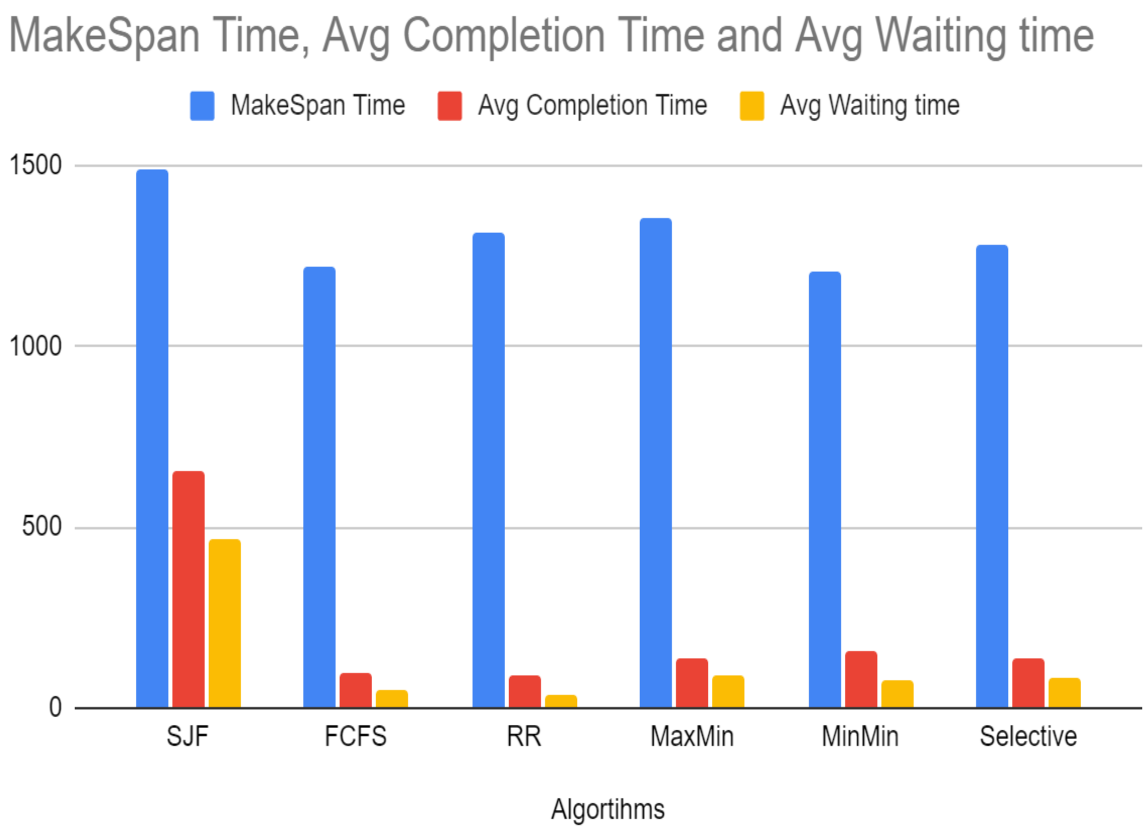


Fig 4.5 Comparsion Graph

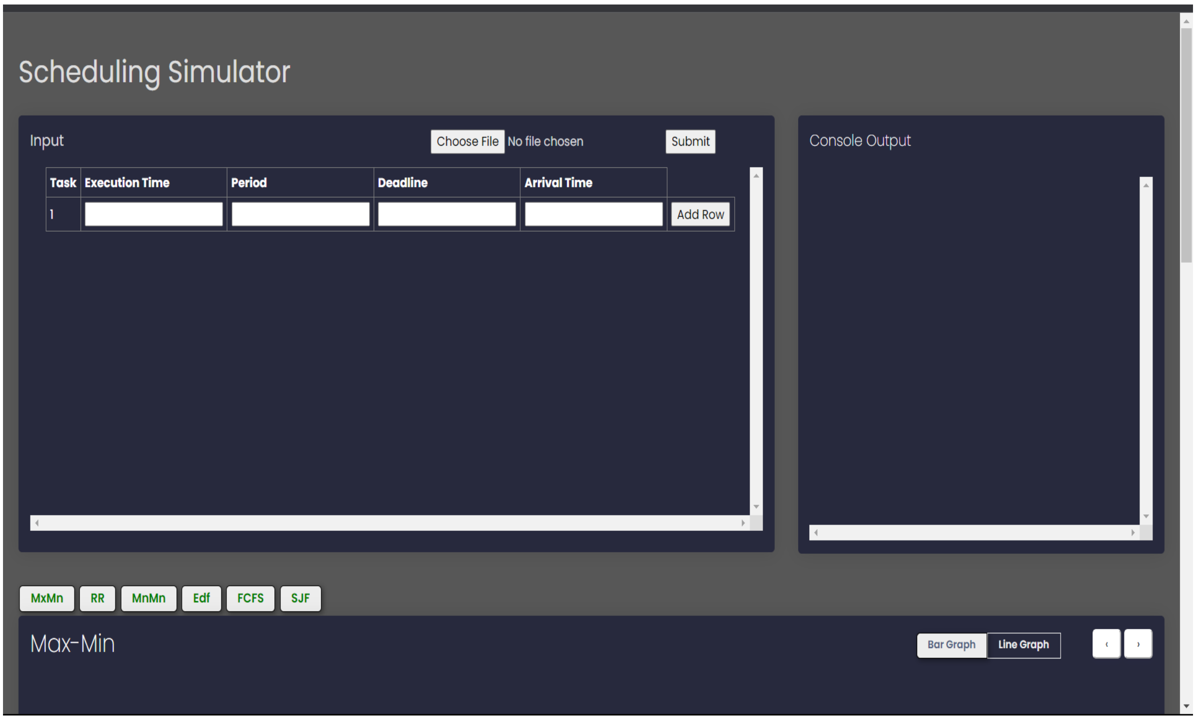
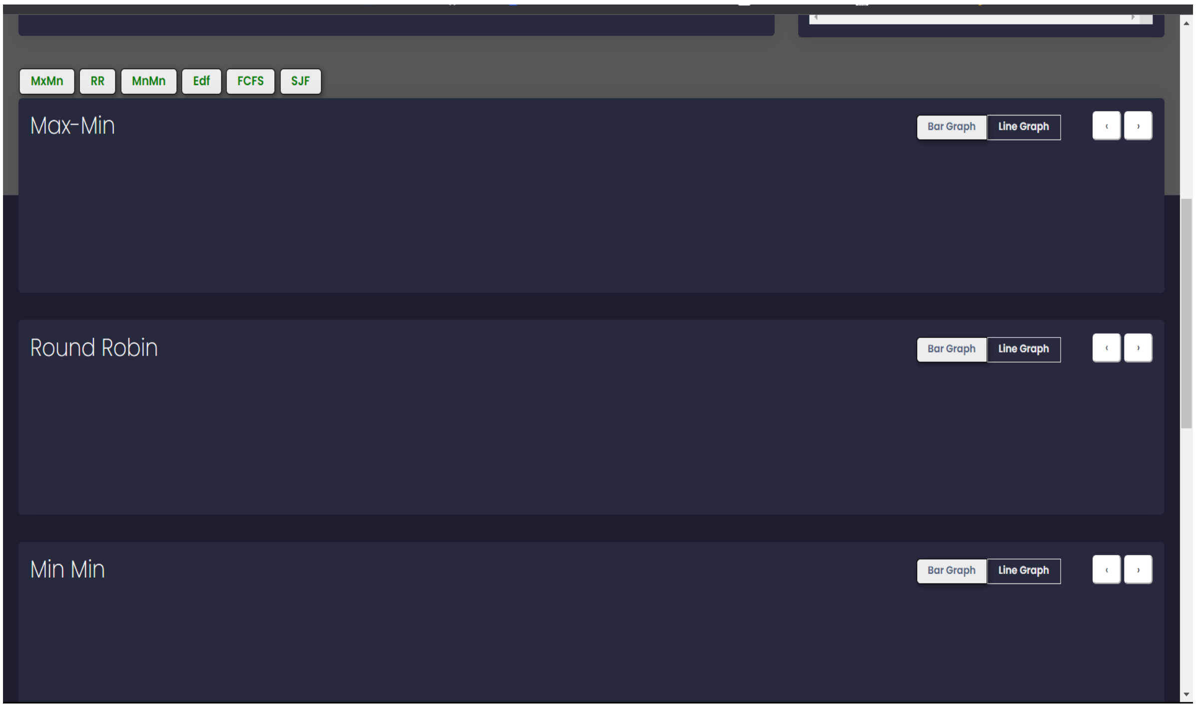


Fig 4.6 Scheduling Simulator



CONCLUSION

In the project, we have introduced the main concepts of schedulings and resources management in the cloud computing. We have presented a comparative analysis of schedulings algortihm by considering the factors waiting time , execution cost ,most important makespan time (execution time).The problem of task scheduling in the Cloud Computing environment is concerned .Maxmin,Minmin,Round Robin are famous algorithms for scheduling tasks on cloud computing environments. Minmin,Maxmin algortihms are scheduled to reduce the make-span time taken , average-waiting time and execution cost. The comparison result indicate that RR,Maxmin,Minmin showed the better results with less overhead.

Future Work

- Future work should examine the impact of additional factors like virtual machines (VMs), data centers, memory, network bandwidth, and storages in cloud environment and reflect those factors in actual physical environments.

- In future works we have planned to improve upon the current algorithms and implement best algorithm in order optimize makespan time , total execution cost.

- We plan to create a webapp to better implementation of your project.

- The project intend to develop the suggested system by extending the practical Cloud platform , such as OpenStack, in order to assess it in a genuine Cloud architecture.
- This work has significant social significance because it lowers modern IT infrastructures' carbon dioxide footprint and energy consumptions in addition to lowering infrastructure and ongoing operating costs.

REFERENCE

[1] Buyya, R., Broberg, J., & Goscinski, A. M.(Eds.). (2010). Cloudcomputing Principles and paradigms (Vol. 87). John Wiley & Sons.

[2] Kumar, M. R., Ganesh, D., & Harish, V.(2017).Various Task Scheduling Algorithms in Clouds.International Journal of Engineeringand Management Research (IJEMR), 7(2),479-485.

[4] Goyal, S. (2014). Public vs. private vs. hybrid vs. community-cloud computing: A critical review. International Journal of Computer Network and Information Security, 6(3), 20.

[5] Kaur, N., & Chhabra, A. (2017). Comparative Analysis of Job Scheduling Algorithms in Parallel and Distributed Computing Environments. International Journal of Advanced Research in Computer Science, 8(3).

[7] Vijay, P., & Anju, B. G. (2014). An Efficient Workflow Scheduling Approach in CloudComputing (Doctoral dissertation).

[8] Grandinetti, L. (Ed.). (2013). Pervasive Cloud Computing Technologies: Future Outlooks and Interdisciplinary Perspectives: Future Outlooks and Interdisciplinary Perspectives. IGI Global.

[9] Karan D. Prajapati, Pushpak Raval ,Miren Karamta,M. B. Potdar Comparison of Virtual Machine Scheduling Algorithms in Cloud Computing Volume 83 – No 15, December 2013.

[10] C. T Lin, “Comparative Based Analysis of Scheduling Algorithms for Resource Management in Cloud Computing Environment”, Vol. 1, Issue-1, July 2013.

[11] Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen “Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment” VOL. 24, NO. 6, JUNE 2013 1107

APPENDIX

```java
package org.cloudbus.cloudsim.examples;
import org.cloudbus.cloudsim.*;

public class SJF_Scheduler {

    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static Datacenter[] datacenter;
    private static double[][] commMatrix;
    private static double[][] execMatrix;

    private static List<Vm> createVM(int userId, int vms) {
        //Creates a container to store VMs. This list is passed to the broker later
        LinkedList<Vm> list = new LinkedList<Vm>();

        //VM Parameters
        long size = 10000; //image size (MB)
        int ram = 512; //vm memory (MB)
        int mips = 250;
        long bw = 1000;
        int pesNumber = 1; //number of cpus
        String vmm = "Xen"; //VMM name

        //create VMs
        Vm[] vm = new Vm[vms];

        for (int i = 0; i < vms; i++) {
            vm[i] = new Vm(datacenter[i].getId(), userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());
            list.add(vm[i]);
        }

        return list;
    }

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

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

        Cloudlet[] cloudlet = new Cloudlet[cloudlets];
```

APPENDIX

```java
package org.cloudbus.cloudsim.examples;
import org.cloudbus.cloudsim.*;

public class SJF_Scheduler {

    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static Datacenter[] datacenter;
    private static double[][] commMatrix;
    private static double[][] execMatrix;

    private static List<Vm> createVM(int userId, int vms) {
        //Creates a container to store VMs. This list is passed to the broker later
        LinkedList<Vm> list = new LinkedList<Vm>();

        //VM Parameters
        long size = 10000; //image size (MB)
        int ram = 512; //vm memory (MB)
        int mips = 250;
        long bw = 1000;
        int pesNumber = 1; //number of cpus
        String vmm = "Xen"; //VMM name

        //create VMs
        Vm[] vm = new Vm[vms];

        for (int i = 0; i < vms; i++) {
            vm[i] = new Vm(datacenter[i].getId(), userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());
            list.add(vm[i]);
        }

        return list;
    }

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

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

        Cloudlet[] cloudlet = new Cloudlet[cloudlets];
```

APPENDIX

```
from django.shortcuts import render

from django.views.decorators.csrf import csrf_protect
from .forms import UserCreationForm
from Main import schedulingAlgorithm

@csrf_protect
def openSimulator(request) :
    # for printing the console into a variable
    old_stdout = sys.stdout
    new_stdout = io.StringIO()
    sys.stdout = new_stdout

    if request.method == 'POST':
        form = UserCreationForm()
        final_array = request.POST.get('final_array')
        interval = request.POST.get('interval')
        noOfTasks = request.POST.get('noOfTasks')
        #form['interval'].initial = interval

        # splitting the input into desired format
        algorithmInputArray = arraySplit(final_array)
        # calculating rate monotonic scheduling algorithm
        rmsArray, edfArray, fcfsArray, sjfArray = schedulingAlgorithm.schedule(algorithmInputArray)
        # formating the input array to send it back to screen
        algorithmInputArray = returnInputArray(algorithmInputArray)
```

```python
        # writing standard output to variable
        outputConsole = new_stdout.getvalue()
        sys.stdout = old_stdout  # setting the standard output back to console

        return render(request, 'index.html',{'sjfArray': sjfArray, 'fcfsArray': fcfsArray, 'edfArray': edfArray, 'rmsArray': rmsArray, 'noOfTasks
    else:
        form = UserCreationForm()
        outputConsole = new_stdout.getvalue()
        sys.stdout = old_stdout  # setting the standard output back to console
        return render(request, 'index.html', {'sjfArray': [], 'fcfsArray': [], 'edfArray': [], 'rmsArray': [], 'noOfTasks': [],'algorithmInputArr


def arraySplit(algorithmInputs):
    algorithmInputs = algorithmInputs[:-1]
    algorithmInputs = algorithmInputs.split("|")
    algorithmInputArray = []
    for algorithmInput in algorithmInputs:
        algorithmInput = algorithmInput[:-1]
        algorithmInputArray.append(list(map(int, algorithmInput.split(",", -1))))
    return algorithmInputArray

def returnInputArray(algorithmInputArray):
    algorithmInputArray.sort(key=lambda x: x[1])
    for inputArray in algorithmInputArray:
        inputArray[1:]
    return algorithmInputArray
```

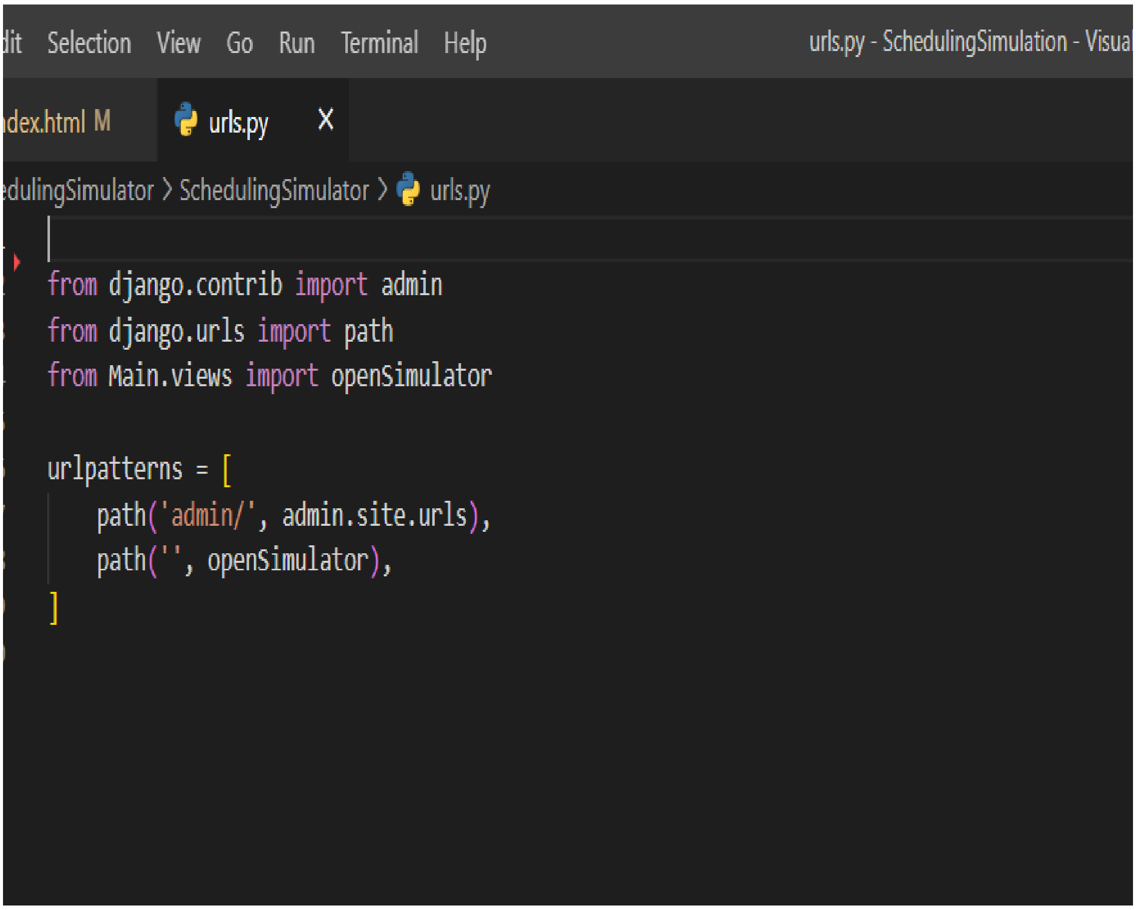
APPENDIX

MaxminBroke... Maxmin.java Selective.java MinMin.java RoundRobinSc... FCFS_Schedul...

```java
package Utils;

public class Constants {
    public static final int NO_OF_TASKS = 50; // number of Cloudlets;
    public static final int NO_OF_DATA_CENTERS = 20; // number of Datacenters;
    public static final int POPULATION_SIZE = 25; // Number of Particles.
}
```

APPENDIX


dit Selection View Go Run Terminal Help
urls.py - SchedulingSimulation - Visual
dex.html M
urls.py X
edulingSimulator > SchedulingSimulator > urls.py
from django.contrib import admin
from django.urls import path
from Main.views import openSimulator
urlpatterns = [
path('admin/', admin.site.urls),
path('', openSimulator),
]

Major Report

ORIGINALITY REPORT

1%	1%	0%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.geeksforgeeks.org Internet Source	1%
2	repo.ust.edu.sd:8080 Internet Source	<1%
3	slideplayer.com Internet Source	<1%
4	Mokhtar A. Alworafi, Atyaf Dhari, Sheren A. El Booz, Suresha Mallappa. "Budget-aware task scheduling technique for efficient management of cloud resources", International Journal of High Performance Computing and Networking, 2019 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		