National College of Ireland

Resource Scheduling for Infrastructure as a Service(IaaS) in cloud computing

MSc Research Project
Cloud Computing

Uday Kumar Das
Student ID: 20191022

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Uday Kumar Das
Student ID:	20191022
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	16/12/2021
Project Title:	Resource Scheduling for Infrastructure as a Service(IaaS) in cloud computing
Word Count:	XXX
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Uday Kumar Das
Date:	16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Resource Scheduling for Infrastructure as a Service(IaaS) in cloud computing

Uday Kumar Das
20191022

Abstract

It is essential to improve user experience and their requirements within SLA limit agreed by cloud providers through resource management. However, both cloud service providers and cloud consumers alike are looking for a well-organized scheduling system in the cloud. In this paper, we will take a look at the challenges of cloud-based Infrastructure as a Service(IaaS) resource scheduling. And to solve these issues, some scheduling algorithms such as BAT, PSO and FCFS are used with QoS parameters like Makespan, Processing Time, response Time, Execution Time and Throughput. With these parameters, a comparative analysis of scheduling algorithms has been done, so that users can utilize the cloud resources more efficiently and effectively. For comparative analysis 5 different combinations of virtual machines and tasks are used for the simulation in CloudSim tool. With this analysis, successfully found that PSO scheduler performing better compaired to other schedulers.

Keywords: IaaS, Resource Scheduling, cloud computing

1 Introduction

In this study, a comparative analysis of cloud computing based on resource scheduling for IaaS is going to be discussed. In order to perform comparisons with different resource scheduling algorithms in cloud platforms, there would be sufficient information collected relevant to the topic. Cloud computing is a process of delivering several resource services with the help of internet service. The accessories used to perform this task are servers, data storage, networking, databases, and software. As a result of introducing cloud computing in the technical field, it gives users easier access to data stored in the cloud database, a fast retrieving process, and this infrastructure is cost-effective.

With the use of cloud computing resources, its users are able to access critical computer resources, such as OS, VMs and online services as well as databases. Cost-saving, energy efficiency, scalability and flexibility are just few of the advantages of cloud computing Kumar et al. (2018). In these discussion benefits of cloud computing can be included, as the cloud-based architecture reduces the use of software and hardware therefore; the cost for purchasing the hardware and software is minimized. Secondly, it provides better performance as compared to the traditional method of computing. In order to resolve resources scheduling issue in IaaS cloud computing, an efficient resource scheduler is needed. In Figure 1 resources scheduling issue in IaaS cloud computing has been shown.

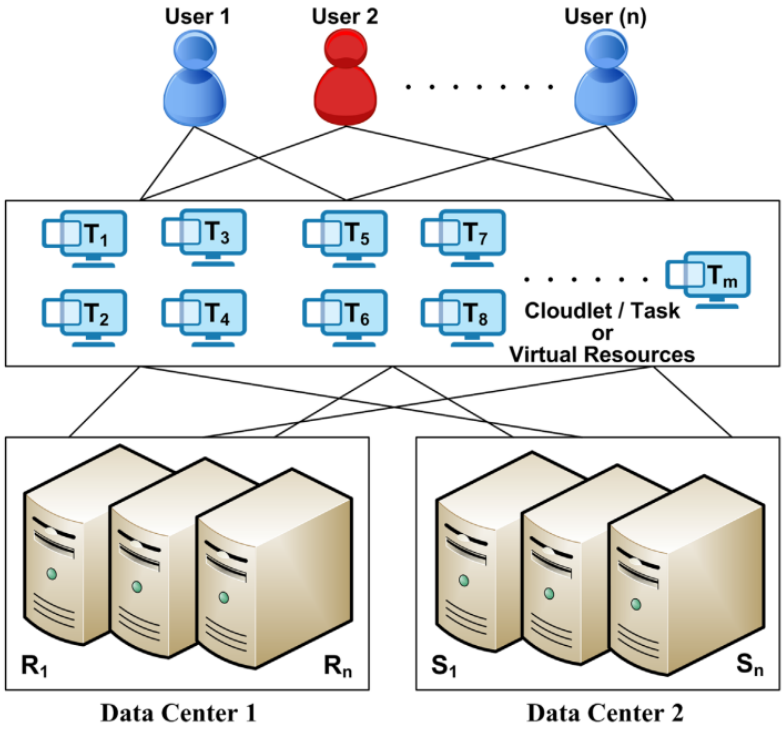

Figure 1: Resources scheduling issue in IaaS cloud computing. Singh & Chana (2016)

Below are the sections discussed in this report: Section 3 describes the Research Methodology for resource scheduling. Section 7 presents the design specification with scheduling architecture. Section 5 presents the implementation of resource schedulers in CloudSim toolkit using JAVA programming language. Section 6 describes the evaluation part where comparative

analysis of schedulers has been shown. Section 7 draws on some important conclusion and future works.

1.1 Justification of the Research

When it comes to Resource Scheduling, there are two key challenges that must be addressed. Uneven load distribution and a lack of ability to effectively utilize resources are two of the most common causes of failure in the workplace. An algorithm of some kind, which can be broadly categorized, is the most effective means of resolving this problem. Particle Swarm Optimization(PSO), BAT and FCFS etc. are all examples of these algorithms. There is still a lot of room for further study in this field, despite the fact that there have already been a number of studies in this area. This led us to build a new method for assigning jobs to virtual machines and comparing it with existing algorithms, which we found to be superior.

1.2 Research Questions

1. How comparative analysis of scheduling algorithms facilitates more efficient and effective cloud resources utilization in IaaS?
2. What are the necessities of resource scheduling for IaaS in cloud computing?

1.3 Research Objective

The objective of this study is given as below,

- To evaluate a comparative study among the existing resource scheduling techniques for IaaS.
- To analyze and find out better algorithm out of the compaired ones considering all the evaluated matrics.

2 Literature review

The study of cloud computing has grown in importance subject for research. So much research has been done for scheduling of cloud computing resources. Many cloud-based resource scheduling methods, as well as three grid-based metaheuristic algorithms, were the subject of a study conducted by Kalra & Singh (2015). For this reason, the majority of the studies have focused on improving the convergence rate and Quality of Service (QoS) by altering the transition operator and initializing a larger sample size before running the metaheuristic algorithm. For example, combining a metaheuristic approach with another algorithm can improve QoS

while also conserving energy. This report also includes observations, optimization metrics, and unresolved topics for future investigation.

Madni et al. (2016*a*) research paper in which they investigate meta-heuristic resource allocation approaches for reducing costs and increasing profits of IaaS cloud users in cloud computing environments has been proposed by researchers. Meta-heuristic algorithms are used to allocate resources in a way that is sensitive to quality of service (QoS) considerations. Matlab, CloudSim, GridSim, and other simulation tools have been utilized, as well as comparative parameters. Meta-heuristic resource allocation algorithms are reviewed and compared in a systematic manner in this research study, which will help researchers in determining future research objectives. These algorithms took into account a slew of ideas for improving the overall performance of meta-heuristics. Cloud computing resource allocation issues can be solved using meta-heuristic algorithms, although none of them are able to provide significantly better performance than other techniques. Resource allocation in the IaaS Cloud was analyzed using meta-heuristic techniques that use execution time to determine resource use.

Gabi et al. (2015) A systematic review of current load balancing solutions in cloud computing has been provided in this research article. Measures for load balancing, resource utilization, throughput, overhead costs, scalability and fault tolerance . There was no way to demonstrate green load balancing of resource scheduling in a single or federated cloud system. However, current approaches ignore studies such as load balancing of electricity use, server consolidation, and virtual machine migration. It is expected that future research will reveal a green multi-goal load-balancing system for heterogeneous cloud environments with high satisfaction of provider advancements. Algorithms and techniques can't solve all the problems in cloud computing, including VM migration and server consolidation; power control and the carbon emission component. Future research aims to uncover efficient multi-goal load balancing of resource scheduling methods for federated heterogeneous and homogeneous cloud environments with high-satisfactory service improvements. Load balancing in cloud computing was also examined by the authors of the paper, which helps to increase the efficiency of cloud data centers while ensuring service level agreement (SLA). In addition, we talked about the various methods of load balancing already in use. In addition, we conducted additional research, including a comparison of our findings with those of other researchers.

According to Strumberger et al. (2019), Cloud computing resource scheduling Hybrid Whale Optimization Algorithms have been suggested as a possible solution. Using swarm intelligence metaheuristics, this study offers a hybridized whale optimization set of rules, which is designed to address the scheduling challenges in cloud environments. In order to more exactly compare the suggested approach's performance, the unique whale optimization was further adjusted for scheduling resources. In order to more accurately compare its total resource performance, the suggested hybrid set of rules was initially tested on a current set of bound-constrained benchmarks. After that, simulations were carried out with real and simulated data sets using different cloud computing resource scheduling strategies. The reliable CloudSim platform had been used to conduct the simulations. In addition to the unique whale optimization, a hybrid set of whale optimization principles was compared against other fashionable metaheuristics and heuristics. Because of this, using a variety of heuristic and metaheuristic approaches, it appears that the proposed hybrid whale optimization set of criteria is superior to the single version on average. Using the proposed set of criteria, improvements were made to the unique whale optimization implementation in cloud computing, as well as improvements in

the scheduling of useful resources.

According to Kumar et al. (2019) increasing expansion of on-demand requests and the varied nature of cloud resources, resource scheduling has emerged as a major concern in cloud computing. Pay-per-use cloud services allow consumers to experience dynamism, uncertainty, and elasticity through the internet. In the last decade, there has been a surge in the number of requests for cloud services. Degradation of service performance or waste of cloud resources may occur as a result of inefficient scheduling approaches that overuse and underutilize resources. Response time, makespan time, availability, consumption, cost, and resource utilization are a few examples of performance indicators that should be considered when developing a scheduling algorithm. Many state-of-the-art scheduling methods based on heuristics, metaheuristics, and hybrids have been documented in the literature to achieve the above-mentioned goal. Scheduling approaches are reviewed and categorized in this study, and their pros and disadvantages are discussed in detail. As a stepping stone for new researchers in the field of cloud computing, we hope our comprehensive review will be useful for the advancement of scheduling technique.

The dynamic distribution of resources is difficult in the cloud environment since there are multiple copies of the same responsibility assigned to different computers. As a result, resource scheduling algorithms that automatically assign assets to responsibilities may be required. Hyper-heuristic scheduling rules are used in the cloud environment in the work written by Nguyen et al. (2019). Using hyper-heuristic methodologies, the mapping of assets is done in an excellent manner. As part of our review of the proposed rules, we compared them to the current set of rules. The suggested set of guidelines reduces both the cost and makespane of computing assets.

2.1 Resource Scheduling Issues in Cloud

In simple words, resource scheduling is the process of determining which tasks and activities must be completed. Cloud computing necessitates that all of the available resources be used effectively. There are several aspects that play a role in the scheduling of resources, including how many resources are required in a typical situation and how many cloud suppliers are available to provide those services. There are a number of factors to keep in mind while deciding how to best allocate cloud computing resources, including as cost, time, quality of service, make-span, and energy consumption.

2.2 Classification of resource scheduling

As shown in the Figure 2 Cost, energy use, efficiency, quality of service, load balancing and utilization are all factors that go into cloud resource scheduling. Provider income, user costs, and provider profit is in the first category. Makespan, Execution Cost, Response Time, Priority, and Execution Time are in the second category. The third area is devoted to energy use. Workload is in the fourth category, on the other hand. It's in the fifth category that you'll find things like availability and fault tolerance as well as throughput, SLA, RTO, Recovery Time and utilization.

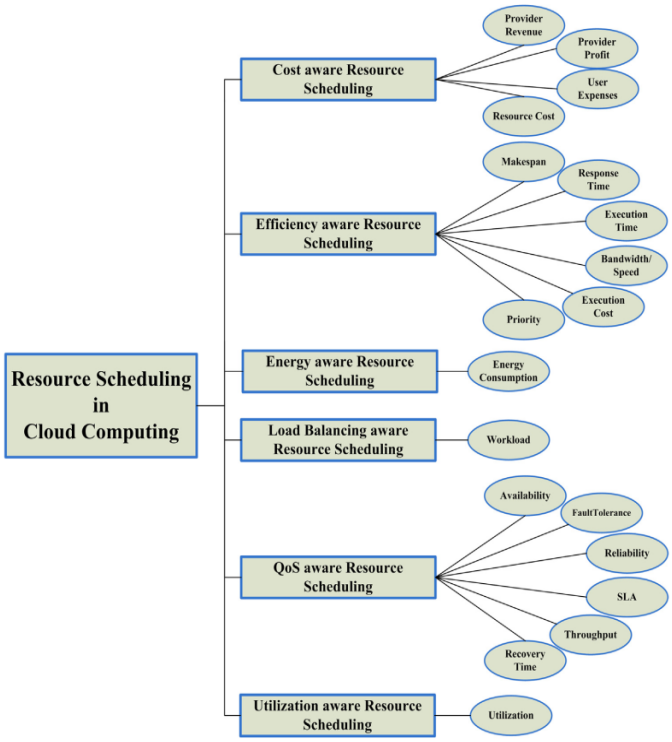

Figure 2: Resources scheduling categories. Madni et al. (2016*b*)

According to cloud providers, reducing energy usage, heat generation, storage, etc., is critical to their business model. While maximizing the utilization of all available resources in order to increase revenue/income. Cloud clients, on the other hand, want the carrier to deliver exceptional overall performance at the lowest possible cost and time . Accordingly, cloud computing studies take into account factors such as cost, energy consumption, execution time, and workloads are some primary considerations.

3 Research Methodology

On the basis of the literature analysis, three scheduling methods have been identified for the development of the suggested model. Those 3 algorithms are BAT, PSO and FCFS . Predefined parameters, such as the number of VMs, number of tasks, number of data centers are needed in order to complete the execution. As a further advantage, both small and big tasks can be completed simultaneously without waiting for other processes to complete while simulation in Cloudsim .

3.1 CloudSim

There are a number of components in CloudSim, including a modeling and simulation toolkit that makes it possible to model and simulate key cloud features, such as a task queue and the

handling of events, as well as the construction of various cloud entities (such as datacenters and datacenter brokers) Calheiros et al. (2009). This simulation tool is written in Java programming language. Below are some CloudSim core features:

- It provides simulated infrastructure to test and create adaptive application provisioning approaches for varied workload and resource mix scenarios.
- Virtualized server hosts, cloud data centers, and energy-aware computing resources can all be modelled and simulated with the help of cloud computing and virtualized server hosts.
- Cloud service brokers, provisioning, and allocation policies can all be modelled using this self-contained framework.
- Allows many virtual services to be created and managed on a single data center node using a single virtualization engine.

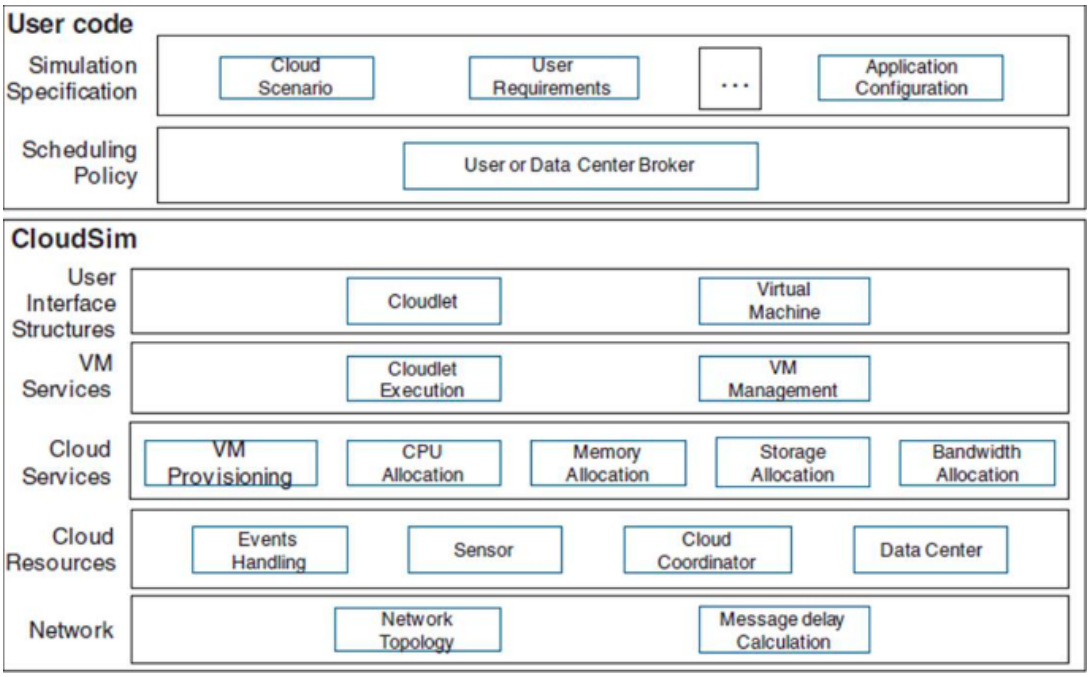

Figure 3: CloudSim Architecture Madni et al. (2016*b*)

This toolkit is used in order to run cloud server hardware model as a software simulation tool, which can then be used to examine its performance under real-world conditions. Because of this, it is evident that as a researcher, the IaaS layer can be simulated. VMs and tasks have their own allocation and scheduling policies, as well as policies for managing power, which may involve VMs migration and consolidation across several hosts, as well as establishing workload parameters for simulation on cloud systems. In Figure 3 CloudSim Architecture has been shown.

3.2 Bat Algorithm

Binary Bat Algorithm is generally based on bat algorithm however it has mammal differences compared to bat algorithm. This algorithm is used to select various binary values and also helps to limit echolocation behaviour of bats, which further helps in performing optimisation Sagnika et al. (2018). In order to develop a binary bat algorithm, modifications in process of position updating, as well as velocity is required. In case of a continuous version of bat algorithm, artificial bats can be moved in space by employing position and velocity vectors. The steps which are considered while developing this algorithm are shown in the Figure 4.

1. Objective function: $f(x), x = (x_1, x_2, x_3, \ldots x_d)t$
2. Initialize bat population x_i and velocity v_i where $i = (1,2,\ldots,n)$
3. Describe pulse frequency f_i at x_i
4. Initialize pulse rate r_i and loudness A_i
5. Whereas (t <maximum number of iterations)
6. Create new solutions by modifying frequency, updating velocities and location.
7. If (rand>c)
8. Select a solution among the best solutions
9. Generate a local solution around the selected best solution
10. End if
11. If (rand <A_i) and $f(x_i) < f(x^*)$
12. Admit new solutions
13. Increase r_i , decrease A_i
14. End if
15. Ranks the bats and find current best x^*
16. End while
17. Display results.

Figure 4: BAT Algorithm Pseudo code Dewangan et al. (2019)

Scheduling of workflow can be depicted as another important aspect as it helps to achieve lower processing time and further helps in maximising effective allocation of cloud resources. In this case, all tasks are divided into bags of tasks (BOT) for different levels. Here, min-max normalization is employed in order to normalise ECT values for different tasks. After that, a dynamic threshold value is employed which further helps in dividing various tasks of a single BOT into two batches. Here a minimum completion time algorithm is employed that helps in scheduling various tasks. Here, the proposed algorithm works by considering two phases for different levels of workflow without assigning priority to tasks.. In the next step, threshold values for each task are calculated after which this value is compared with maximum normalized value. It is done here in order to segment each task into various small as well as large batches. Following process is carried out from starting node to end node by employing top to bottom fashion. In next step, tasks are assigned to VMs by selecting tasks with a minimum time of execution. Following step is continued until the scheduling of various tasks. In addition to this, in second phase, EFT, as well as EST, is calculated for takes as it is a prerequisite

before scheduling tasks. The steps which are considered while developing this algorithm are as follows:

3.3 PSO Algorithm

PSO is a bionic intelligent optimization method inspired by animal swarm behavior. It is also a stochastic optimization technique based on swarm intelligence that optimizes by utilizing the information exchange mechanism of bird group individuals Masdari et al. (2017). Through collaboration amongst individuals, this algorithm can recall both personal and global best data. Each particle in the population represents a potential solution to the optimization problem. In Figure 5 Pseudo code for PSO algorithm has been shown.

1. Initialization:

   1.1 Find the initial population by clustering method;
   1.2 Calculate the objective function of initial solutions (**f**);
   1.3 Set initial velocity vector equal to zeros;
   1.4 Set Pb=f, Gb=min(f); (Gb is Gbest and Pb is Pbest)

2. Repeat the following steps until the stopping condition is met:

   (a) Update Gb and Pb:
   For i=1:PS (Population Size)
   If f_i<Pb(i)
   Pb(i)= f_i
   End
   End
   b=min(f); (**b** is the best solution of current generation)
   If b<Gb
   Gb=b;
   End

   (b) Generate next population by using (2) and (3)

   (c) Checking the feasibility of generated solutions and repair of them by described Strategy.

   (d) Calculate the objective function of generated solutions (**f**)

3. Report the best solution

Figure 5: PSO Algorithm Pseudo code Ghaderi et al. (2012)

3.4 FCFS Algorithm

There's a popular scheduling method known as First Come First Served (FCFS) and as its name suggests, it prioritizes processes based on their order of arrival. Like the FIFO Queue data structure, where the first item to be put to the queue exits first, First Come First Serve follows the same logic M & Jayavel (2018). Typically, Batch Systems employ this type of technology. With the use of a Queue data structure, a new process can be added, and the scheduler can select the most relevant one based on how long it's been in the queue.

If a necessary resource is not available in FCFS, The system would then merely wait for the resource to become available, but this algorithm would distribute it incrementally or, to put it another way, Wait for the next request to be serviced before moving on to the next one. It adheres to a dynamic allocation in the direction of Depending on the current usage constraint may apply. Data is then allocated based on the requests. It's up to you which category the request falls into. if a request falls short of a certain threshold in terms of time constraints, cost constraint-based consideration when it is above the service's value threshold. allocation. The request that gives the maximum value within the constraints of cost Priority is given to cost-cutting measures, and so on.

4 Design Specification

The main goal of Scheduling is primarily concerned with making the most efficient use of available resources and reducing the amount of time required to complete each task. Scheduling activities should be done in such a way that the Quality of Services (QoS) is improved while retaining efficiency and fairness among the tasks. As a result, the number of tasks that may be transmitted to the cloud is increased, which improves the system's overall performance.

4.1 Scheduling Architecture

In Figure 6 Scheduling Algorithm has been shown.

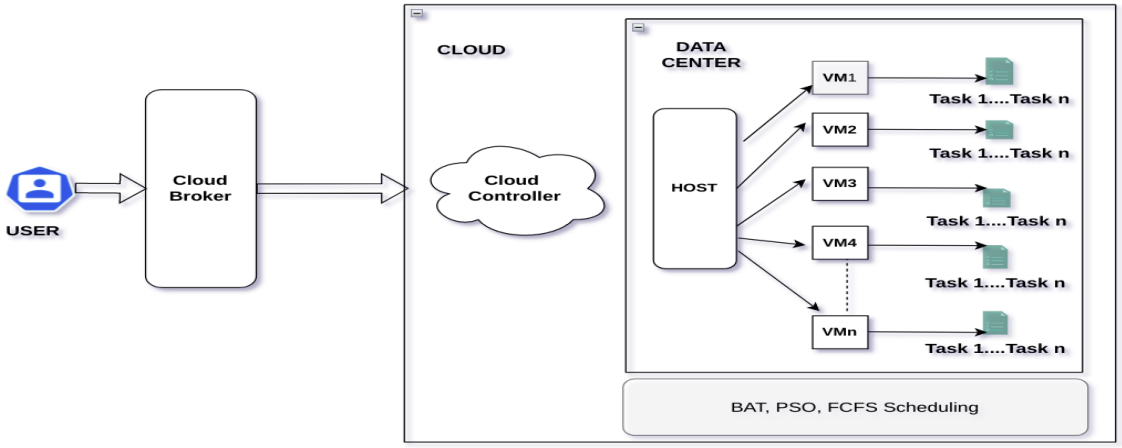

Figure 6: Scheduling Architecture.

User tasks are submitted to the DataCenterBroker, which acts as a dispatcher between the user and the DataCenter and helps to schedule work on virtual machines. A number of hosts in the DataCenter are used to schedule virtual machines, and the DataCenterBroker schedules tasks on those virtual machines in accordance with the policies of the DataCenterBroker. The number of Data Center Brokers is comparable to that of cloud consumers. To schedule tasks, the Data Center Broker communicates with the cloud controller.

In simulation, some of the most common classes used are:

- **Cloud Broker**: In a cloud computing system, one of the levels is comprised of brokers. Data centers and cloud users work together through a broker, who acts as an intermediary. VMs and Tasks are commonly created in Cloudsim. Tasks and Vms are retrieved from users, then created and sent to datecenters for processing.
- **Datacenter**: Modeling the core hardware of every cloud environment, which is the Datacenter. In addition to allowing for the specification of the Datacenter's functional needs, this class also includes methods for configuring VM resource allocation policies.
- **VM**: Data members for bandwidth, RAM, mips (million instructions per second), and VM size are provided, as are setter and getter methods for these parameters in the VM class, which represents a virtual machine.
- **Cloudlet**: Tasks like processing, memory access, and file updating all fall under cloudlet class. A job's length, size, and are all stored in this class, which exposes methods identical to those in the virtual machine class while also describing the execution time, status of a given work.
- **DatacenterBroker**: A DatacenterBroker is a third-party agent acting on behalf of a client. In order for VMs to work properly, it must be in charge of all parameters of VM formation He et al. (2013).

4.2 System requirements

As shown in the Figure 7 Cloudsim Parameters Used for Resource Scheduling has been shown. For the specification we need JAVA programming language to be installed on Linux or Windows PC with 2.50 GHz and 10.0 GB RAM. These algorithms will then be applied to 5 test scenarios, which will include both application operations and network types. Randomly generated tasks and nodes are established by the program with a minimum of 30 tasks and six nodes, after which more tasks and nodes can be assigned as needed. All networks are connected to each other via a high-speed bandwidth. We can use this to figure out things like makespan, energy consumption, throughput and execution time etc We can obtain the average value for each task by repeating this process three times.

4.3 CloudSim Parameters Used for Simulation

As shown in the Figure 8 Cloudsim Parameters Used for Resource Scheduling has been shown. For the simulation, few parameters are required like number of VMs, number of tasks, RAM, MIPS, Bandwidth, number of Data Centers, VMM etc.

SPECIFICATION	VALUE
Simulator Tool	CloudSim 3.0
Programming Language	JAVA 11.0
Min RAM Required	10 GB
Bandwidth	2.5 GHz
No of Min Nodes	6-100

Figure 7: Design Specification.

PARAMETRS	VALUE
Total no of VMs	10-100
Total no of Tasks	30-300
MIPS	500-2000
RAM (VM)	1024 MB
Bandwidth	500-1000
No of Data Center	5
VMM	Xen
System Architecture	x86
OS	Linux

Figure 8: CloudSim Parameters Used for Resource Scheduling.

5 Implementation

In order to run the simulation, utilized CloudSim (Version 3.0) which is accessible on GitHub and requires only a few simple pre-requisites. JAVA programming language needed to be installed in order to get started. Installed the Eclipse IDE, which enables Java developers to write code and run CloudSim simulations. By importing CloudSim library in Eclipse IDE, now developer can write the code as per the requirement. Virtual machines of three different sizes and all feasible combinations have been used to evaluate the BAT, PSO , and FCFS schedulers outputs for this study. Each of the three methods that contain the suggested algorithm is tested for a total of five possible combinations. Using this method of task execution, the quickest tasks are handled by a single pool of virtual machines, while the most time-consuming ones are routed to a different pool of VMs. Used BAT, PSO and FCFS schedulers, so that the comparison between these can be shown in graphs and tables. And also with the help of comparative analysis of parameters like Makespan, Total execution time etc.

Below are some Classes, Packages and Methods used for the simulation:

- **BAT Package:** BAT algorithm based Cloudlet Scheduling.
- **PSO Package:** Particle Swarm Optimization(PSO) based Cost Effective Cloudlet Scheduling.
- **FCFS Package:** FCFS or FIFO based Cloudlet scheduling.
- **Cloudlet:** Each class that has to perform a task should implement Cloudlet interface. NullPointerExceptions can be avoided by utilizing the NULL object rather than assigning null to Cloudlet variables.
- **vmList:** It is a collection of operations on lists of VMs on cloud.
- **Datacenter:** This class allows the specification of the Datacenter's functional needs, this class also includes methods for configuring VM resource allocation policies.
- **createVM**: It is a method which can create VMs/Cloudlets and send it to a specific broker.
- **DatacenterCharacteristics:** The attributes and rules for a data center are defined by the this Interface.
- **UtilizationModel:** It is an interface which controls resources usage by a Cloudlet.
- **GenerateMatrices:** It is a class which is used to generate the matrices used internally to set latency and bandwidth between elements.

Experiments are conducted in CloudSim to determine variables such as the processing time, make span time, and processing costs.

6 Evaluation

Using Cloudsim, a set of experiments are carried out to verify the suggested algorithm's efficacy. There are minimum 10 VMs which is assigned for minimum 30 tasks. Using BAT, PSO, and FCFS, these findings are then compared quantitatively to those obtained using other similar VM combinations. Various characteristics like makespan, processing time, response time, execution time and throughput are then computed for comparative analysis of resource scheduling in CloudSim.

6.1 Parameter Analysis

Below are the details of QoS parameters used for comparative analysis between BAT, PSO and FCFC schedulers:

- **Makespan**: To calculate the make span time, add up all the time spent on each task, counting backwards from the time when the first one began to the time till the last task execution. Makepan time can be calculated using the following formula:

$$Makespan = LastCloudletFinishTime$$

- **Processing Time**: It is the amount of time required by an algorithm to complete a particular task. The following formula is used to compute it:

$$ProcessingTime = Cloudlet_{len}/Vm_{mips} * PES_{count}$$

- **Response Time**: This is the amount of time it takes for the system to respond to a command and begin the task at hand. the time difference between when a job starts and when it takes up actual CPU time; for example, the time when the task first arrived.

$$ResponseTime = ActualCpuTimeOfCloudlet - ExecutionStartTimeOfCloudlet$$

- **Execution Time**: The amount of time it takes the system to complete the task. It is the difference between the task's completion time and its start time. Narwal (2020).

$$ExecutionTime = FinishTimeOfCloudlet - ExecutionStartTimeOfCloudlet$$

In Figure 9 BAT Scheduler Simulation Output Using CloudSim tool has been shown. In Figure 10 BAT Scheduler code sample in Eclipse IDE has been shown.

6.1.1 Case Study 1: Comparison of Processing Time Between Schedulers

Here the Processing Time is compared for BAT, PSO, and FCFS schedulers in terms of performance. The processing time of the scheduling algorithms BAT, PSO, and FCFS with 300

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    05         SUCCESS         03            03       815.03       00.1          815.13
    12         SUCCESS         06            06       980.91       00.1          981.01
    00         SUCCESS         02            02      1055.15       00.1          1055.25
    03         SUCCESS         02            02      1310.98       00.1          1311.08
    11         SUCCESS         04            04      1366.13       00.1          1366.23
    01         SUCCESS         05            05      1520.91       00.1          1521.01
    22         SUCCESS         03            03      1035.23       815.13          1850.36
    09         SUCCESS         06            06      1990.93       00.1          1991.03
    02         SUCCESS         05            05      2058.18       00.1          2058.28
    21         SUCCESS         03            03      2128.54       00.1          2128.64
    17         SUCCESS         02            02      1459.39       1311.08          2770.47
    13         SUCCESS         06            06      1812.03       981.01          2793.04
    04         SUCCESS         02            02      1778.89       1055.25          2834.13
    08         SUCCESS         04            04      2840.5        00.1          2840.6
    07         SUCCESS         05            05      934.37        2058.28          2992.64
    23         SUCCESS         03            03      1333.16       1850.36          3183.51
    24         SUCCESS         03            03      1431.66       2128.64          3560.3
    06         SUCCESS         05            05      2621.01       1521.01          4142.02
    26         SUCCESS         02            02      1558.15       2834.13          4392.28
    25         SUCCESS         02            02      1725.63       2770.47          4496.1
    10         SUCCESS         05            05      1593.88       2992.64          4586.52
    14         SUCCESS         04            04      3409.72       1366.23          4775.95
    16         SUCCESS         04            04      2740.68       2840.6          5581.28
    29         SUCCESS         02            02      1104.16       4496.1          5600.26
    15         SUCCESS         06            06      3719.47       1991.03          5710.5
    19         SUCCESS         05            05      1293.26       4586.52          5879.78
    28         SUCCESS         05            05      200.28        5879.78          6080.06
    20         SUCCESS         04            04      1439.09       4775.95          6215.04
    18         SUCCESS         05            05      2089.44       4142.02          6231.46
    27         SUCCESS         02            02      2501.86       4392.28          6894.14
Makespan using BAT: 4258.358085640115
Total CPU Time  51848.593333333345
Total Response Time: 6894.04
This simulation is for 10 VMs and 30 Tasks.
BAT.BATscheduler finished!
```

Figure 9: BAT Scheduler Simulation Output Using CloudSim

Console | BATscheduler.java

```
206         double response_time = 0.0;
207         double waiting_time = 0.0;
208         double makespan = calcMakespan(list);
209         dft.setMinimumIntegerDigits(2);
210         for (int i = 0; i < size; i++) {
211             cloudlet = list.get(i);
212             Log.print(indent + dft.format(cloudlet.getCloudletId()) + indent + indent);
213             if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
214                 Log.print("SUCCESS");
215
216                 CPU_time += cloudlet.getActualCPUTime();
217                 response_time = cloudlet.getFinishTime() - cloudlet.getSubmissionTime();
218                 waiting_time += cloudlet.getWaitingTime();
219                 Log.printLine(indent + indent + dft.format(cloudlet.getResourceId()) +
220                         indent + indent + indent + dft.format(cloudlet.getVmId()) +
221                         indent + indent + dft.format(cloudlet.getActualCPUTime()) +
222                         indent + indent + dft.format(cloudlet.getExecStartTime()) +
223                         indent + indent + dft.format(cloudlet.getFinishTime()) +
224                         indent + indent + (dft.format(cloudlet.getWaitingTime())));
225             }
226         }
227
228         Log.printLine("Makespan using BAT: " + makespan);
229         Log.printLine("Total CPU Time  "+ CPU_time);
230         Log.printLine("Total Response Time: " + response_time);
231         Log.printLine("Avg. Waiting Time: " + (waiting_time / Constants.NO_OF_TASKS));
232         Log.printLine("This simulation is for "+ Constants.NO_OF_VM + " VMs and " + Constants.NO_OF_TASKS + " Tasks.");
233     }
234
235     private static double calcMakespan(List<Cloudlet> list) {
236         double makespan = 0;
237         double[] dcWorkingTime = new double[Constants.NO_OF_DATA_CENTERS];
238
239         for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
240             int dcId = list.get(i).getVmId() % Constants.NO_OF_DATA_CENTERS;
241             if (dcWorkingTime[dcId] != 0) --dcWorkingTime[dcId];
242             dcWorkingTime[dcId] += execMatrix[i][dcId] + commMatrix[i][dcId];
243             makespan = Math.max(makespan, dcWorkingTime[dcId]);
244         }
245         return makespan;
246     }
247
248
249 }
```

Figure 10: BAT Scheduler code sample in Eclipse IDE

Tasks and 100 VMs is 56940.1000, 38869.9100 and 56077.9833 milliseconds respectively. These schedulers processing time has been shown in the Figure below 11:

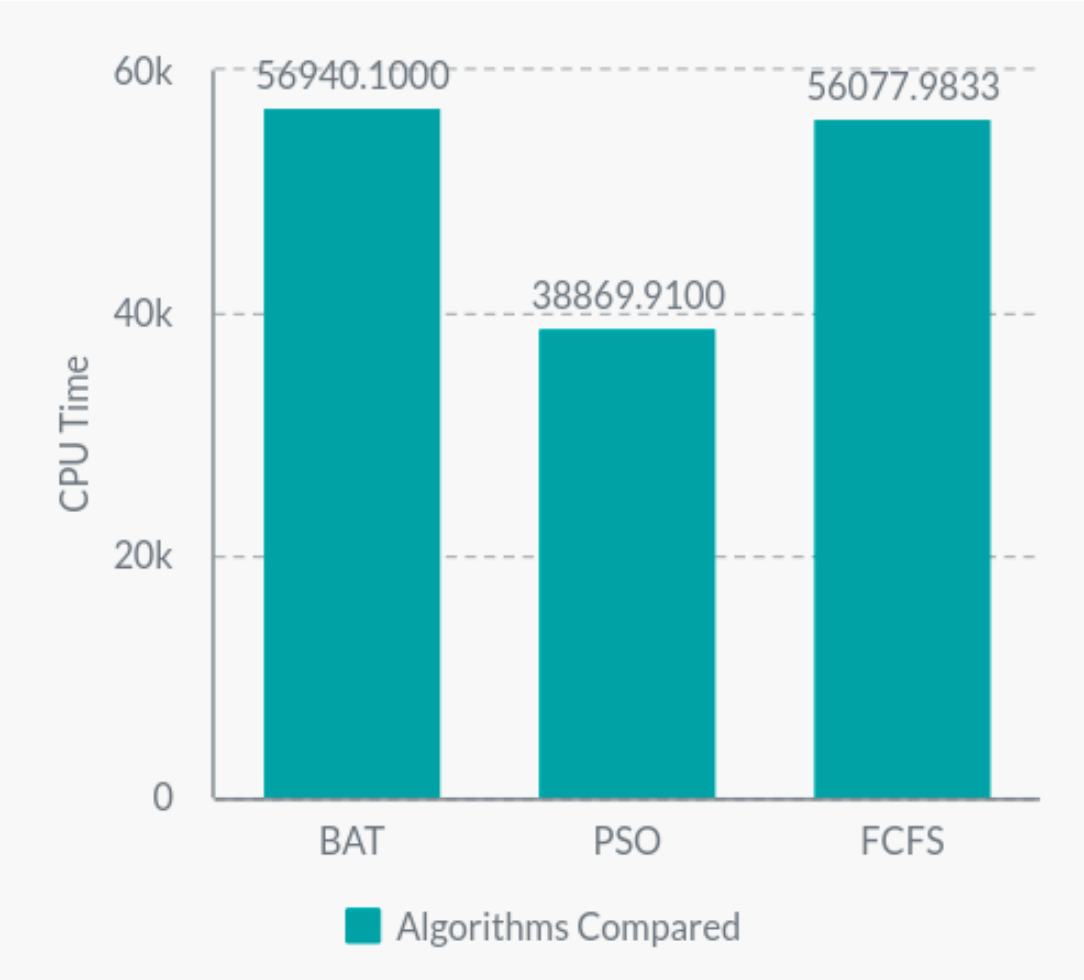

Figure 11: Comparison of BAT, PSO and FCFS w.r.t Processing Time.

According to the graph, it is obvious that the suggested PSO Scheduler works best when compared to BAT and FCFS schedulers.

6.1.2 Case Study 2: Comparison of Makespan Time Between Schedulers

Here the Makespan Time is compared for BAT, PSO, and FCFS schedulers in terms of performance. The Makespan time of the scheduling algorithms BAT, PSO, and FCFS with 30 Tasks and 10 VMs is 5365.3528, 2578.9159, and 4900.7489 milliseconds, respectively.It takes 12324.0215, 3272.4633 and 6767.7581 millisecond to process 300 tasks and 100 virtual machines for BAT, PSO, and FCFS.In the Figure 12, CloudSim simulation results for Makespan Time has been shown.

In the Figure 15, Comparison of BAT, PSO and FCFS w.r.t Makespan Time has been shown.

No of VMs	No of Tasks	BAT	PSO	FCFS
10	30	5365.3528	2578.9159	4900.7489
20	60	6085.9958	2658.0872	4151.7935
30	100	9925.8191	3190.0227	5372.7518
50	150	11170.8686	2795.4950	4125.5288
100	300	12324.0215	3272.4633	6767.7581

Figure 12: Simulation Output for Makespan Using CloudSim

According to the graph, it is obvious that the suggested PSO Scheduler works best when compared to other schedulers.

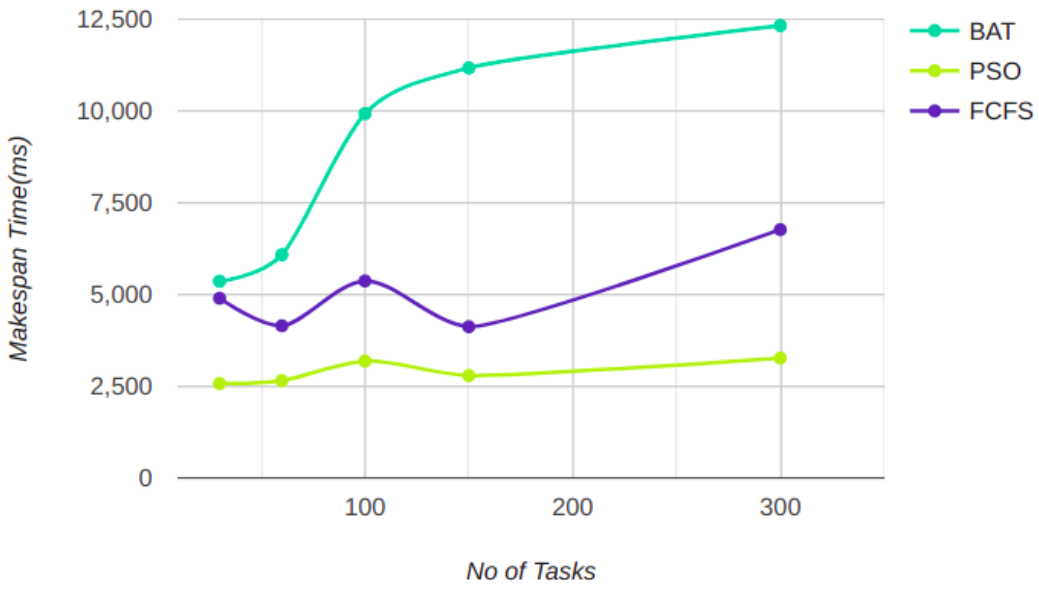

Figure 13: Comparison of BAT, PSO and FCFS w.r.t Makespan Time.

6.1.3 Case Study 3: Comparison of Response Time Between Schedulers

Here the Response Time is compared for BAT, PSO, and FCFS schedulers in terms of performance. The Response time of the scheduling algorithms BAT, PSO, and FCFS with 30 Tasks and 10 VMs is 8438.8166, 4867.54, and 8827.4733 milliseconds, respectively. It takes 8523.8300, 6366.1000 and 5886.4933 millisecond to process 300 tasks and 100 virtual machines for BAT, PSO, and FCFS.In the Figure 14, CloudSim simulation results for Response Time has been

shown.

No of VMs	No of Tasks	BAT	PSO	FCFS
10	30	8438.8166	4867.54	8827.4733
20	60	9080.7200	4259.7833	6244.1600
30	100	9314.3366	4164.2833	1993.1299
50	150	10303.6133	5320.2833	3146.3333
100	300	8523.8300	6366.1000	5886.4933

Figure 14: Simulation Output for Response Time Using CloudSim

In the Figure 15, Comparison of BAT, PSO and FCFS w.r.t Response Time has been shown.

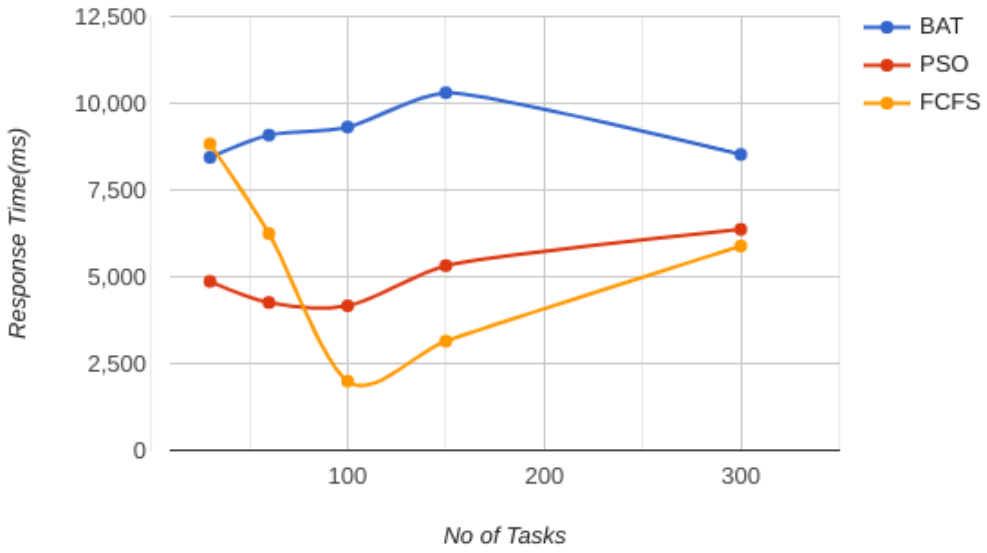

Figure 15: Comparison of BAT, PSO and FCFS w.r.t Response Time.

6.1.4 Case Study 4: Comparison of Average Waiting Time Between Schedulers

Here the Response Time is compared for BAT, PSO, and FCFS schedulers in terms of performance. The Response time of the scheduling algorithms BAT, PSO, and FCFS with 30 Tasks and 10 VMs is 2297.0171, 1360.7245, and 2491.4806 milliseconds, respectively. It takes

5510.7460, 4059.0265 and 5003.3161 millisecond to process 3000 tasks and 100 virtual machines for BAT, PSO, and FCFS.In the Figure 16, CloudSim simulation results for Average Waiting Time has been shown.

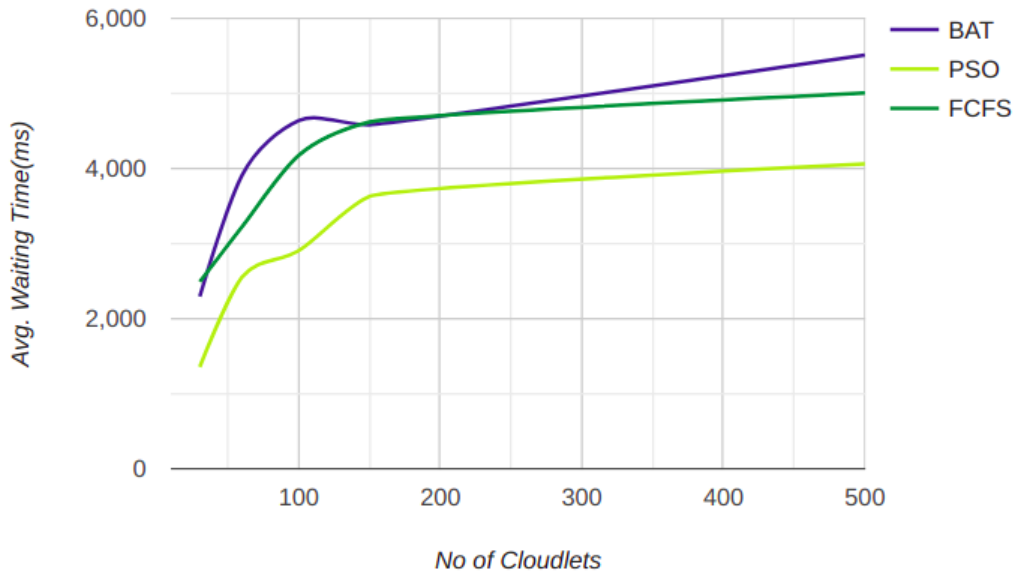

Figure 16: Comparison of BAT, PSO and FCFS w.r.t Avg. Waiting Time.

6.2 Discussion

For this reason, the criteria listed here were chosen to be analyzed:

- The cost of the VMs configuration is determined by the number of execution cycles necessary to perform the given set of activities on it. As a result of this optimization, this number should be as less is feasible.
- QoS parameters are important for cloud user experience. Using task size sorting, we were able to significantly cut the average waiting time. As a result, the cloud's capacity to retain users is determined by the amount of time it takes for the cloud to be deployed.
- After analyzing all evaluated matrics like Makespan, Processing Time, Response Time and Execution Time, found out that PSO scheduler is better than BAT and FCFS.

7 Conclusions and Future Work

In this paper various scheduling methods has been examined in this work. BAT, PSO and FCFS algorithms have been recognized as effective scheduling of cloud resources in literature review.

Inefficient use of resources has resulted in several issues, including an increase in execution time and costs. Specifically, the jobs based on the scheduling process are routed to pools for execution. When compared to BAT, PSO and FCFS algorithms, PSO algorithm really reduces the wait time with all given CloudSim parameters , therefore PSO scheduling reducing overall wait time. Another advantage of the suggested approach is that it reduces both the cost and the time required to complete jobs.

For this simulation, used total of 10 to 100 VMs and 30 to 300 Tasks for execution with 5 different combinations. The ideal VM combination for each method is determined based on the evaluation of the most important QoS metrics like Makespan, Processing Time, Response Time, Execution Time. According to the results obtained, the PSO algorithm outperforms the BAT and FCFS algorithms in terms of waiting time, and also in terms of execution cost and completion times. In future research, dynamically allocation of cloud resources by a novel scheduling algorithm will be purposed which will be depending on load balancing and network bandwidth.

References

Calheiros, R. N., Ranjan, R., De Rose, C. A. & Buyya, R. (2009), 'Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services', *arXiv preprint arXiv:0903.2525* .

Dewangan, B. K., Agarwal, A., Venkatadri, M. & Pasricha, A. (2019), 'Self-characteristics based energy-efficient resource scheduling for cloud', *Procedia Computer Science* **152**, 204–211. International Conference on Pervasive Computing Advances and Applications- PerCAA 2019.
URL: *https://www.sciencedirect.com/science/article/pii/S1877050919306969*

Gabi, D., Ismail, A. S. & Zainal, A. (2015), 'Systematic review on existing load balancing techniques in cloud computing', *International Journal of Computer Applications* **125**(9).

Ghaderi, A., Jabalameli, M., Barzinpour, F. & Rahmaniani, R. (2012), 'An efficient hybrid particle swarm optimization algorithm for solving the uncapacitated continuous location-allocation problem', *Networks Spatial Economics - NETW SPAT ECON* **12**, 1–19.

He, Z. T., Zhang, X. Q., Zhang, H. X. & Xu, Z. W. (2013), Study on new task scheduling strategy in cloud computing environment based on the simulator cloudsim, *in* 'Advanced Materials Research', Vol. 651, Trans Tech Publ, pp. 829–834.

Kalra, M. & Singh, S. (2015), 'A review of metaheuristic scheduling techniques in cloud computing', *Egyptian informatics journal* **16**(3), 275–295.

Kumar, A., Kumar, R. & Sharma, A. (2018), 'Equal: energy and qos aware resource allocation approach for clouds', *Computing and Informatics* **37**(4), 781–814.

Kumar, M., Sharma, S., Goel, A. & Singh, S. (2019), 'A comprehensive survey for scheduling techniques in cloud computing', *Journal of Network and Computer Applications* **143**, 1–33.
URL: *https://www.sciencedirect.com/science/article/pii/S1084804519302036*

M, P. & Jayavel, K. (2018), 'Iot based visualization of weightage based static task scheduling algorithm in datacenter', *International Journal of Engineering Technology* **7**, 439.

Madni, S. H. H., Latiff, M., Coulibaly, Y. & Abdulhamid, S. M. (2016*a*), 'An appraisal of meta-heuristic resource allocation techniques for iaas cloud', *Indian Journal of Science and Technology* **9**(4), 1–14.

Madni, S. H. H., Latiff, M. S. A., Coulibaly, Y. & Abdulhamid, S. M. (2016*b*), 'Resource scheduling for infrastructure as a service (iaas) in cloud computing: Challenges and opportunities', *Journal of Network and Computer Applications* **68**, 173–200.
URL: *https://www.sciencedirect.com/science/article/pii/S1084804516300674*

Masdari, M., Salehi, F., Jalali, M. & Bidaki, M. (2017), 'A survey of pso-based scheduling algorithms in cloud computing', *Journal of Network and Systems Management* **25**(1), 122–158.

Narwal, A. (2020), 'Credit based scheduling with load balancing in cloud environment', *International Journal of Advanced Trends in Computer Science and Engineering* **9**, 1121–1127.

Nguyen, T., Nguyen, B. M. & Nguyen, G. (2019), Building resource auto-scaler with functional-link neural network and adaptive bacterial foraging optimization, *in* 'International Conference on Theory and Applications of Models of Computation', Springer, pp. 501–517.

Sagnika, S., Bilgaiyan, S. & Mishra, B. S. P. (2018), Workflow scheduling in cloud computing environment using bat algorithm, *in* 'Proceedings of first international conference on smart system, innovations and computing', Springer, pp. 149–163.

Singh, S. & Chana, I. (2016), 'A survey on resource scheduling in cloud computing: Issues and challenges', *Journal of grid computing* **14**(2), 217–264.

Strumberger, I., Bacanin , N., Tuba, M. & Tuba, E. (2019), 'Resource scheduling in cloud computing based on a hybridized whale optimization algorithm', *Applied Sciences* **9**(22).
URL: *https://www.mdpi.com/2076-3417/9/22/4893*