

# TASK SCHEDULING ALGORITHM IN CLOUD COMPUTING BASED ON MODIFIED ROUND ROBIN ALGORITHM

<sup>1</sup> RUBA ABU KHURMA, <sup>2</sup> HEBA AL HARAHSHEH, <sup>3</sup> AHMAD SHARIEH

<sup>1</sup>Ph.D Student, University of Jordan, Department of Computer Science, King Abdullah II School for Information Technology, Amman, Jordan

<sup>2</sup>Ph.D Student, University of Jordan, Department of Computer Science, King Abdullah II School for Information Technology, Amman, Jordan

<sup>3</sup>PROF., University of Jordan, Department of Computer Science, King Abdullah II School for Information Technology, Amman, Jordan

E-mail: <sup>1</sup> ruba\_abukhurma@yahoo.com, <sup>2</sup> Heba.moh.h@gmail.com, <sup>3</sup> sharieh@ju.edu.jo

## ABSTRACT

Cloud computing offers opportunities to access remote physical and virtual resources. Due to the continuing development of cloud computing, many challenges face this technology. One of these challenges is tasks scheduling. It refers to the process of allocating users' tasks to virtual machines (VMs) with a goal of minimizing the turnaround time and improving the resource utilization. Tasks scheduling is considered NP hard problem with  $O(m^n)$  run time complexity to schedule  $n$  tasks on  $m$  resources. The process of tasks scheduling consumes a large solution space and with lacking of algorithms that can find the optimal solution in a polynomial run time.

This paper presents a review study of various task scheduling algorithms in cloud environment including: RR, MaxMin, MinMin, FCFS, MCT, PSO, and GA, with a case study on modified round robin (MRR) algorithm. The MRR algorithm has been tested using CloudSim toolkit. The results show that when using the MRR algorithm to schedule a number of Cloudlets over a number of VMs, the average waiting of run time becomes less than when using RR in the same environments. Thus, it is advisable to use the proposed MRR for tasks scheduling in cloud computing, because it reduces the average waiting time and keeps the good features of the RR such as fairness, avoiding starvation, based on simple rule, dynamic based on CC environment situations, and suitable for load balancing.

**Keywords:** *Cloud Computing, CloudSim, Dynamic Scheduling, Modified Round Robin, Task scheduling, Virtual Machines.*

## 1. INTRODUCTION

This study investigates the performance of various jobs scheduling algorithms under cloud computing environment, namely: Round Robin (RR), Maximum-Minimum (MaxMin), Minimum-Minimum (MinMin), First Come First Service (FCFS), Minimum Completion Time (MCT), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). Furthermore, a case study about modified round robin (MRR) algorithm and the relation between RR and MRR algorithms will be described.

With the advancement in the information technology (IT) industry, several computing paradigms have been presented including the High Performance Computing (HPC), Parallel Computing (PC), Distributed Computing (DC), Cluster Computing (CIC), Grid Computing (GC), Mobile Computing (MC) and Cloud Computing (CC) [1]. The CC is an internet based computing model to provide on demand services to clients such as security, virtualization, web infrastructure, Web 2.0 and other developing technologies [2]. The idea is to pool shared and configurable resources such as servers, storage, platforms and applications to be accessed over the internet.

In CC, the commercial service provider like Microsoft Azure, Amazon EC2, Google apps engine, etc offers the utility “pay as you go” to rent services to clients and the resources are delivered as virtual machines based on service level agreement (SLA) and negotiation between the service provider and the consumer[3]. The access mechanism for resources on the distributed network follows common internet protocols and standards of networking and the resources. The processing and storing for data are done via private cloud owned by a company or using a public cloud and third part server in one of the distributed datacenters. There are also other emerging cloud deployments types such as hybrid, community, and federated [4].

CC delivers three different service models which are categorized, as depicted in Fig1, into: infrastructure as a service (IaaS), platform as a service (PaaS), and software as services (SaaS) [5]. In IaaS, the resources, storage and processing power are provided in the form of virtual machines. PaaS provides a computing platform which includes operating systems, programming languages, and web based applications. In SaaS, the software and database access is allowed for a user according to a particular usage based payment model.



Figure1: Cloud Services [6]

Nowadays, CC becomes one of the most significant and attractive technology trend in the IT market since it efficiently provides dynamic, flexible, reliable, sustainable, scalable and self-managed computing infrastructure [7]. Moreover, there is no need to purchase new hardware, to pay for training practices or to license new programs. So, the cost saving is a major benefit for both small and large enterprises which allow them to focus on

innovation instead of being trapped in hardware and software setup. On the other hand, flexibility is considered a bonus for cloud computing with varying hardware configurations, different platforms, operating systems and many software packages. The usefulness of cloud computing also is represented by offering suitable levels of availability, reliability and fault tolerance which guarantee the continuity of cloud functions and the provision of services even if some cloud sites are down.

The ubiquitous growth of CC faces many obstacles and challenges such as security, performance, and resource management [8]. Task scheduling is one of the main issues related to resource management and has curious impact on the efficiency, throughput, and resource utilization in a cloud environment. Task scheduling in cloud computing concerns with assigning users’ tasks to the available resources in a way the system utilization and throughput are improved and SLA requirements are not violated [9].

Tasks scheduling is a matter of mapping a stream of users’ tasks into the available resources in cloud computing environment. This is an optimization problem, since the scheduler tries to find the optimal tasks-VMs mapping (best matching) with regard to scheduling times such as the response time, make span and completion time.

Assume that there are K tasks  $T = \{T_1, T_2, \dots, T_i, \dots, T_K\}$  and N resources  $R = \{R_1, R_2, \dots, R_j, \dots, R_N\}$  in the current system of cloud computing. Here, cloud resources refer to the virtual resources [10].

The main target is to minimize the total processing and waiting times associated with scheduling in a way the system throughput is maximized and quality of service (QoS) constraints are preserved which include many user input constraints, as in formulas (1) and (2).

$$\text{Minimize (Processing time)} = \text{Min} (\sum_{i=1}^k (T_i.\text{length}/\text{VMmips}) * \text{NO\_PE}) \quad (1)$$

Where  $T_i.\text{length}$  is the length of the submitted task,  $\text{VMmips}$  is the number of instructions (in millions) per second, which measures the CPU speed, and  $\text{PE}$  is the number of processing elements in a VM.

$$\text{Minimize (Waiting time)} = \text{Min} ((\sum_{i=1}^k (T_i.\text{waiting-time}())) \quad (2)$$

Tasks scheduling steps are modeled as illustrated in Figure2.

1. A user of CC submits a task to a scheduler.
2. A scheduler communicates with Cloud Information System (CIS) for getting information about resources.
3. CIS provided the resources information to the scheduler.
4. The scheduling algorithm does its role for mapping task to the suitable resource and
5. submits the task to the winner resource (decision process for allocating a resource).
6. The user gets the identification (id) of the resource and uses it through cloud interface.
7. The scheduler gets over time updated information about the status of a cloud to manage the schedule.
8. The information is sent to the user.

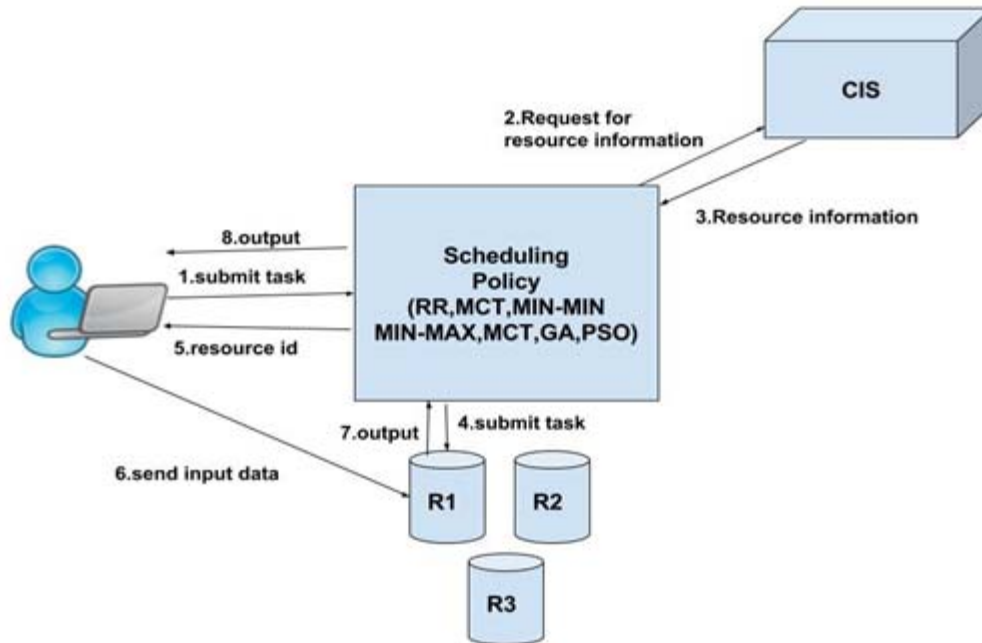


Figure2: Steps of task scheduling in cloud computing

The process of allocating virtual machines fairly among tasks is to minimize the workload. Execution time is considered complicated especially because there are many influencing parameters that should be taken into consideration like task completion time, cost, response time, power consumption etc [11]. In the other side, the task scheduling problem is considered non polynomial (NP)-hard problem, hence there is a requirement for finding a suitable optimization approach to solve the problem in a polynomial time [12].

Task scheduling in CC is a challenge issue. There are needs to improve the performances and quality of services, and reduce cost of execution. The task scheduling in CC should consider the benefits of both the users and the service providers. The main research question, here, is: does the modified RR algorithm will achieve better performance for task scheduling in CC? This paper presents an

investigation of exiting strategies to handle the challenge issues and attempts to enhance the round robin algorithm for better benefits.

The rest of the paper is organized as follows. The next section presents a literature review about different task scheduling algorithms in CC. Section 2 discusses some of the scheduling algorithms in CC. Section 3 describes the methodology of modified round robin. Section 4 shows a description for Cloud Sim Toolkit. Section 5 briefs the experimental setup, results and discussion. Conclusion is discussed in Section 6.

## 2. LITERATURE REVIEW

In this section, a brief review study is presented which shows some of the most relevant research works done for enhancing the performance of scheduling tasks in cloud computing environment. The review includes various tractable algorithmic

solutions that have been presented according with their chronological order.

Tracing the literature back to 2010, Van den Bossche, R., Vanmechelen, K, and Broeckhove, J. [13] proposed an optimal tasks scheduling policy in hybrid cloud model. The hybrid is composed of both private and public cloud. The scenario of the problem is that some workloads have to be outsourced from private cloud to public cloud during the peak load intervals where there are no sufficient resources in private cloud that cover the submitted users tasks. These workloads are constrained by a deadline and QoS requirements. In this case, a decision making process is needed to select which workloads to outsource and to what cloud provider, in such a way the utilization in the internal data center is maximized and the cost of running the outsourced tasks is minimized. A linear programming technique was used to tackle this optimization problem and it performed well in terms of cost minimization, feasibility and scalability.

In 2011, Sindhu, S. and Saswati Mukherjee [14] proposed two algorithms for scheduling tasks in cloud computing based on the processing requirements of a task and the computational capacity of a resource. The first algorithm, named Longest Cloudlet Fastest Processing Element (LCFPE), tries to minimize the makespan (the total time for executing all tasks) by assigning the lengthier cloudlets (tasks) to a Processing Elements (PEs) having high computational power. The second algorithm, named Shortest Cloudlet Fastest Processing Element (SCFP), does the opposite. In SCFP, the shorter cloudlets are mapped to PEs having high computational power process. Using this algorithm, the flow time is minimized and the starvation of longer jobs is avoided. They suggested for future work experimenting more algorithms that use heuristic methods and also to consider the priority of tasks.

Ant Colony Optimization algorithm (ACO) was used for optimizing scheduling under cloud [15]. ACO resembles the ant colony behavior where an ant moves in random direction searching for food sources. In case of task scheduling, the tasks are analogues to ants and the virtual machines imitate the food sources. ACO is used to solve combinatorial optimization problems with several targets of performance and costs. The complexity analysis of ant colony optimization scheduling in cloud with  $K$  tasks and  $R$  resources is computed based on two stages: firstly, the algorithm finds the optimal path in  $O(K)$  time complexity. Secondly,

the optimization judgment is made to meet the cost and performance constraints in  $O(KN)$  where  $N$  is the number of tasks. So, the overall complexity of the algorithm is  $O(KN)$ . The space complexity of this algorithm is  $O(1)$ , since the algorithm consumes constant number of tasks and resources and does not involve any dynamic variables. Gogulan, R., A. Kavitha, and U. Karthick Kumar, in 2012, presented a new nature inspired algorithm called Multiple Pheromone Algorithm (MPA) which belongs to ACO algorithms [16]. MPA generates dynamic schedule so the task is completed in minimum time and the resource utilization is enhanced. MPA achieved better QoS than ACO and algorithms according to three studied parameters: makespan, cost and reliability constraints.

Other study was performed by Ravichandran, S. and E. R. Naganathan, in 2013 [17]. They applied genetic algorithm to solve the problem of uncertainty in tasks arrival to the cloud. This problem results in tedious binding for tasks to VMs. The proposed idea to solve this problem was dynamic scheduling where arrived user's tasks are queued and the scheduler role is to sort them based on the computation and memory usage; then, GA is used to pick each task and find the best fit for allocating a task to available virtual machines and obtain the global optimization.

In 2014, Agarwal, Dr. and Saloni Jain [18] presented in their work, for task scheduling in CC, new algorithm, named generalized priority algorithm (GPA). The algorithm was experimented and compared with FCFS and RR algorithms for varying number of VMs and workload traces and using CloudSim simulator. The results show that the proposed algorithm was more efficient than FCFS and RR algorithms.

A multi-objective tasks scheduling algorithm for mapping tasks to VMs was proposed by Lakra, Atul Vikas, and Dharmendra Kumar Yadav in 2015 [19]. As opposed to single criteria based algorithm which considers execution time only, the multi-objective task scheduling algorithm takes into consideration some other QoS parameters like execution time, cost, bandwidth of user etc. The algorithm was evaluated using CloudSim and the results showed improved throughput for the datacenter and reduced cost without violating the SLA.

In 2015, A. Moradbeiky and V. Bardsiri [20] conducted a research using Cuckoo Optimization based Task Scheduling Algorithm. In this



algorithm, bird nets simulate the processing units (virtual machines) and eggs are the tasks. The cuckoo's role is to lay eggs (tasks) in the nets (VMs). Using this method and based on the number of virtual machines and the number of tasks (inputs of the algorithm) various orders of these machines are examined each time until tasks are allocated to hosts in the right manner.

One year later, in 2016, Hamad, Safwat A., and Fatma A. Omara presented an improved genetic algorithm (TS-GA) for task scheduling problem in the cloud computing environment [21]. The aim of the proposed algorithm was to minimize the completion time, and maximize resource utilization. The results was simulated using CloudSim simulator and the results showed reduced cost, improvement in resource utilization, increased speedup, and higher ratio for algorithm efficiency when compared with default GA and RR algorithms.

Dandhwani, Vanita, and Vipul Vekariya presented new K-mean based task scheduling algorithm [22]. The idea of the proposed algorithm is to create clusters of tasks using k-mean clustering technique, and then allocates clusters to VMs as per capacity of VMs. The results showed reduced execution time and makespan and improved the total cloud system performance.

Recently, in 2017, Madni, Syed Hamid Hussain [23] compared the performance of six rule based heuristic algorithms for tasks scheduling based on some parameters like cost, degree of imbalance, makespan and throughput using CloudSim. These algorithms are First Come First Serve (FCFS), Minimum Completion Time (MCT), Minimum Execution Time (MET), MaxMin, MinMin and Suffer age. The MinMin algorithm performed better than other heuristics. They recommended that heuristic algorithms used as a standard to compare new proposed algorithms. They suggested, for future work, comparing of heuristic and meta-heuristic algorithms may give optimal results and cover the loopholes of each other to achieve the optimization of task scheduling in cloud computing and to improve the MinMin algorithm for optimizing the cost for task scheduling in cloud computing.

From this review, it is clear that there is not a beneficial strategy to optimize task scheduling in CC considering all issues and parameters. For examples, some consider the utilization in the internal data center and the cost of running time [12]; cost minimization, feasibility and scalability

[13]; minimization flow time and avoiding starvation [14]; solving combinatorial optimization problems with several targets of performance and costs [15]; and better QoS according to makespan, cost and reliability constraints parameters[16]. Recent study [21] was simulated using CloudSim simulator and the results showed reduced cost, improvement in resource utilization, increased speedup, and higher ratio for algorithm efficiency when compared with default GA and RR algorithms [21].

We choose to investigate the performance of the modified RR algorithm for task scheduling in CC for several reasons. These include the RR fairness in allocation the processes to CPUs, avoiding starvation [35], efficiency with respect to average waiting time; and modifying the RR algorithm to take online decisions and dynamically adjust the time slice based on situations of the CC environment is a more viable alternative for the standard RR and promising results regarding the response time.

## 2.1 Selected Task Scheduling Algorithms

In this study, seven common task scheduling algorithms are analyzed under cloud computing environment: RR, MaxMin, MinMin, FCFS, MCT, PSO, and GA. Table 1 and Table2 describe each job scheduling algorithm.

## 3. MODIFIED ROUND ROBIN

The round robin algorithm is one of the best CPU scheduling algorithms that achieved the fairness in allocating the processes to CPU based on time quantum granted to each process [35]. It forces each running process to be preempted from CPU to ready queue so that no process allocate the CPU for a long time and so no starved processes in the system. This algorithm has been applied also within the cloud computing environment for allocating resources and proved its efficiency with respect to average waiting time for each process [36]. If the number of tasks is  $N$  in a specific CC deployment, then each task will allocate  $QT=1/N$  of the virtual machine processing time and it will wait no more than  $(N-1)*QT$ , and the scheduling overhead (selecting task for execution) is  $O(1)$  [37]. However, the CC is not stable environment as there are many changes may arise while the tasks are waiting in the ready queue; and depending on a fixed time slice approach may not lead always to the best system performance. On the other hand, determining the size of the time quantum and the ratio of context switch needs a careful thought

(Context switch is nearly close to 10% of the time quantum time) [38].

A new enhancement was proposed in the literature to cope with this bottleneck in front of round robin algorithm. Modifying the round robin algorithm to take online decisions and dynamically adjust the time slice based on the situations is a more viable alternative for the standard RR and promising results regarding the response time. In

this study, we will present the modified round robin (MRR) algorithm and find the relation between the results of running MRR and RR [39]. Figure 8 shows the task scheduling based on dynamic quantum time. MRR calculates the time quantum dynamically so that a task may be granted a quantum time which differs from the quantum time granted to other task; for example task1(T1) has a  $QT_1=5$  while task(T2) quantum time  $QT_2=9$ .

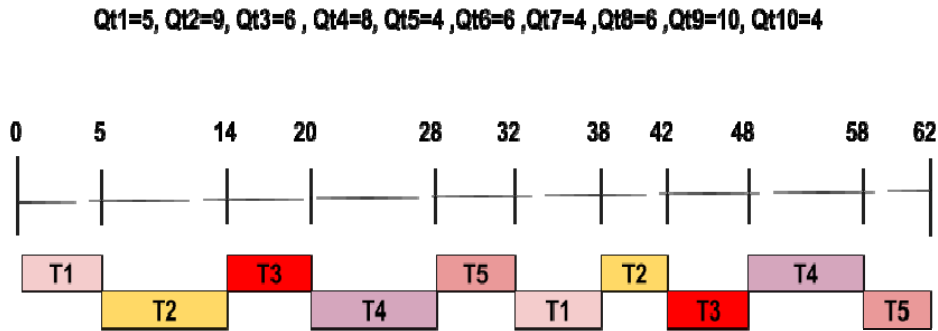


Figure 8: Dynamic task scheduling using MRR

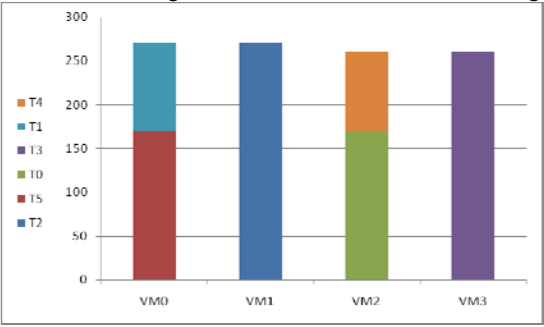
Table 1: Information about Job Scheduling Algorithms

Scheduling Algorithm	Description	Pros	Cons
RR [24]	It is a pre-emptive algorithm that distributes the jobs on the available VMs in a round form (cyclic manner), where the jobs are stored in a ring queue. Each job is allocated a quantum of time and if it can't complete within its turn, then it will be interrupted and stored back in the tail of queue and wait for its next turn. The algorithm repeats until each task in the queue is being assigned to at least one virtual machine.	<ol style="list-style-type: none"> <li>1- No need for a preprocessing step to fetch the nominated VM.</li> <li>2- Distribute the load equally among VMs.</li> <li>3- Focuses on fairness among the scheduled tasks.</li> <li>4- Jobs are executed in turn and never waiting for previous job to finish execution (starvation free).</li> <li>5- The scheduler will not wait until all processing power of a VM is exhausted before it moves to next VM.</li> <li>6- It is based on a simple rule.</li> </ol>	<ol style="list-style-type: none"> <li>1- Long jobs take longer time to complete execution.</li> <li>2- Servers may be overloaded</li> <li>3- Preemptive policies depend on the length of time slice and case on short time slice this will cause many switching.</li> </ol>
MaxMin [25,26]	The algorithm computes the time completion for each task on all VMs and dispatches the largest task(maximum completion time) and assigns it to fastest machine (the one with minimum completion time for that particular task).The algorithm is repeated until all tasks are exhausted.	<ol style="list-style-type: none"> <li>1- Reduce the waiting time of long tasks so they never starved.</li> <li>2- The utilization is increased.</li> <li>3- The response time is minimized.</li> <li>4- The makespan is reduced since smaller jobs are executed concurrently while other longer jobs are executed.</li> </ol>	<ol style="list-style-type: none"> <li>1- As it first selects the large tasks for execution the smaller tasks are delayed.</li> <li>2- Not effective in load balancing.</li> </ol>
MinMin [27]	The algorithm computes the time completion for each task on all VMs and dispatches the	<ol style="list-style-type: none"> <li>1- Smaller makespan, since tasks are scheduled on the fastest machines where they are completed earlier.</li> </ol>	<ol style="list-style-type: none"> <li>1- It increases the total completion time of all the tasks and hence increases</li> </ol>

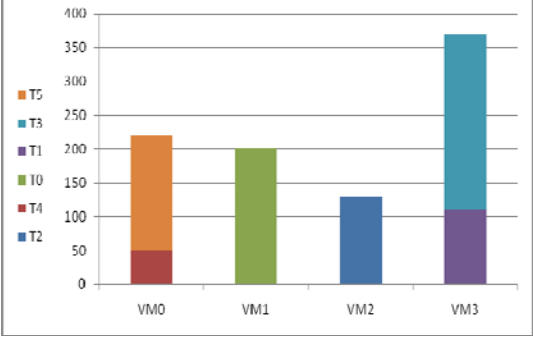
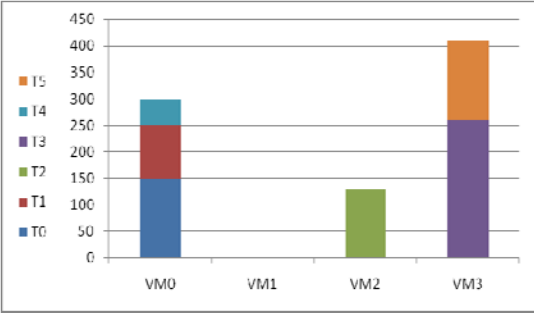
	smallest task (smallest completion time ) and assigns it to fastest machine (the one with minimum completion time for that particular task).The algorithm is repeated until all tasks are exhausted.	2- The algorithm is operative for the task scheduling in CC. 3- No pre-checking on machines load; it just smaller tasks on faster machines. Increase the throughput.	the makespan (Batched jobs completion time). 2- The long tasks have to wait for smaller tasks to end their execution. Unbalanced load.
FCFS [28]	The incoming task aims the queue with smallest waiting time. The queue is managed by FIFO mechanism (first come first out).	1- Simple. 2- Easy to understand	1-Non preemptive. 2-The short jobs at the back of a queue will wait until long task in the front of queue is completed. 3-It is based on single criterion for scheduling.
SJF [29]	Shortest Job First Scheduling (SJN) is used to order a set of tasks by placing the shorter task in the front of the queue and the longer tasks at the end of a queue.	1- Reduce average waiting time because it reduces the waiting time of the short jobs and increases the waiting time of the long jobs.	1- Can result in starvation for longer jobs when there are a large number of small jobs.
MCT [30]	The algorithm scans the available VMs to find the most appropriate machine to assign a job for .The VM is selected based on the minimum completion time by taking into consideration the processing speed and the current workload on a machine.	1-MCT considers both execution times and resource loads so it is considered a successful heuristic that could be implemented in CC .	1-The process of assigning a task to certain machine with minimum completion time is done in arbitrary order so each time a task is assigned to the fastest machine in the remaining resources pool.
PSO [31]	PSO is a type of meta-heuristics algorithms which applies self-adaptive global search for optimization and it starts with random initialization for position and velocity for the practices population. Referring to a problem of task scheduling, the tasks are considered the particles and the number of tasks in the workflow is the dimension of these particles. Each dimension has a value associated to it indicates the resource where the tasks workflow is heading to. So the mapping between tasks and resources is represented by a particle in PSO. Like GA each particle is evaluated using fitness function.	1- The traffic workload using PSO is balanced. 2- Scalable as it could be used with any number of tasks and resources. 3- It can find near optimal solutions for mapping all tasks in the workflow to the set of available resources. 4- Less use of mathematical operators compared with GA and consequently less need for parameters tuning. 5- Simple and effective to be used in wide applications with little computation overhead compared to GA.	1- Easy to fall into local optimum in large search space. 2- Slow convergence.
GA [32,33,34]	Using GA for tasks scheduling, each chromosome represents the job vector and the tasks are the positions in this vector. The content of each position (task) is the id for the machine that the task assigned to. The population depicts several mappings for tasks to machines and the GA plays the role of	1- The algorithm can find the near optimal solution (the fittest mappings ) since it considers in each generation the past and the new solutions to formulate the best scheduling. 2- Apply stochastic search to deal with large space problems. 3- Lessen the waiting time. 4- Handle local optima problem.	1- Complexity in computations and long time requirement. 2- Trial/error parameters such cross over and mutation percentages.

	<p>performing heuristic search to find the optimal solution (mapping). The fitness function in this case measures the quality of solution (is the execution time for all tasks). Generally the algorithm imitates the mechanism of natural selection strategy which include the four steps: selection, cross over, mutation and evaluation</p>		
--	--	--	--

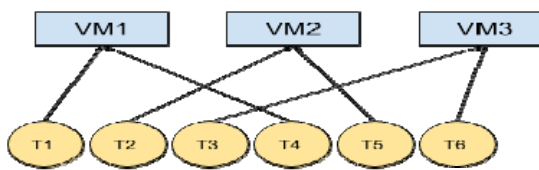
Table 2: Job Scheduling Algorithms Pseudo Codes and Examples

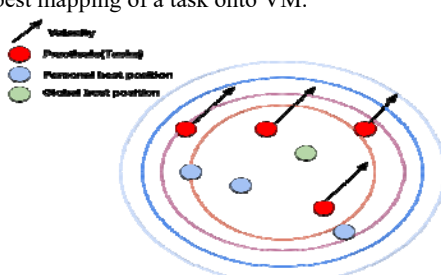
Algo rithm	Pseudo code	Example																																			
Max Min	<p>// T: #tasks, M: #VMs, Cij : completion time of a task, Eij: execution time of a task i, Rj: ready time of task i on virtual machine j</p> <ol style="list-style-type: none"> <li>1. For i=1 to T</li> <li>2. For j=1 to M</li> <li>3. Cij=Eij + Rj End for End for</li> <li>4. Do until all the un schedule tasks are exhausted</li> <li>5. For each unscheduled task</li> <li>6. Find the minimum completion time of the task and virtual machine that obtains it End for</li> <li>7. Find task tp with maximum completion time</li> <li>8. Assign task tp with maximum completion time</li> <li>9. Assign task tp from pull of unscheduled tasks</li> <li>10. Update ready time of the machine that gives the maximum completion time End do</li> </ol>	<p>Assume a cloud computing with 4 machines and six tasks as shown in Table 2.1.</p> <p style="text-align: center;">Table2.1:Tasks execution time on various VMs</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Tasks/ machines</th> <th>VM0</th> <th>VM1</th> <th>VM2</th> <th>VM3</th> </tr> </thead> <tbody> <tr> <td>T0</td> <td>150</td> <td>200</td> <td>170</td> <td>250</td> </tr> <tr> <td>T1</td> <td>100</td> <td>120</td> <td>140</td> <td>110</td> </tr> <tr> <td>T2</td> <td>250</td> <td>270</td> <td>130</td> <td>310</td> </tr> <tr> <td>T3</td> <td>350</td> <td>330</td> <td>300</td> <td>260</td> </tr> <tr> <td>T4</td> <td>50</td> <td>70</td> <td>90</td> <td>110</td> </tr> <tr> <td>T5</td> <td>170</td> <td>200</td> <td>230</td> <td>150</td> </tr> </tbody> </table> <p>The tasks will be assigned to the machines as shown in Figure3.</p>  <p style="text-align: center;">Figure3:MinMax tasks scheduling</p>	Tasks/ machines	VM0	VM1	VM2	VM3	T0	150	200	170	250	T1	100	120	140	110	T2	250	270	130	310	T3	350	330	300	260	T4	50	70	90	110	T5	170	200	230	150
Tasks/ machines	VM0	VM1	VM2	VM3																																	
T0	150	200	170	250																																	
T1	100	120	140	110																																	
T2	250	270	130	310																																	
T3	350	330	300	260																																	
T4	50	70	90	110																																	
T5	170	200	230	150																																	
Min	<p>// M:#tasks, N:#VMs ,Cij: completion time of a task, Eij: execution time of a task, Rj: ready time of task i on virtual machine j</p> <ol style="list-style-type: none"> <li>1. As steps 1-3 in MaxMinDo until all the unscheduled tasks are exhausted</li> <li>2. For each unscheduled task</li> </ol>	<p>Assume the same cloud computing environment in the previous example. The tasks will be assigned to the machines according to MinMin as shown in Figure 4.</p>																																			



<p>Min</p>	<ol style="list-style-type: none"> <li>3. Find the minimum completion time of the task and virtual machine that obtains it End for</li> <li>4. Find task <math>t_p</math> with earliest completion time</li> <li>5. Assign task <math>t_p</math> with earliest completion time</li> <li>6. Assign task <math>t_p</math> from pull of unscheduled tasks</li> <li>7. Update ready time of the machine that gives the minimum completion time. End do</li> </ol>	 <p>Figure4:MinMin tasks scheduling</p>																																								
<p>MCT</p>	<ol style="list-style-type: none"> <li>1. Steps 1-3 as in MaxMin Do until all the unscheduled tasks are exhausted</li> <li>2. For each unscheduled task Find the minimum completion time of the task and virtual machine that obtains it. End for</li> <li>3. Find task <math>t_p</math> with earliest completion time</li> <li>4. Assign task <math>t_p</math> with earliest completion time</li> <li>5. Assign task <math>t_p</math> from pull of unscheduled tasks</li> <li>6. Delete task <math>t_p</math> from pull of unscheduled tasks Update ready time of the machine that gives the minimum completion time. End do</li> </ol>	<p>Assume the same cloud computing environment in the previous example. The tasks will be assigned to the machines as shown in Figure 5.</p>  <p>Figure5:MCT tasks scheduling</p>																																								
<p>FCFS</p>	<ol style="list-style-type: none"> <li>1- Place each incoming Task at the end of the service queue.</li> <li>2- The first task in the queue is assigned to VM when it is available until the end of its execution time</li> </ol>	<p>Assume a cloud computing environment consists of 10 tasks and 3 VMs.</p> <p>Table2.2: Task time calculation on various VMs using FCFS</p> <table border="1" data-bbox="634 1318 1390 1927"> <thead> <tr> <th>Tas k Id</th> <th>Arri val time</th> <th>Exec ution time</th> <th>Comple tion time</th> <th>Waiting time each on VMs</th> <th>Assi gne d VM</th> <th>Waiting queue on each VM</th> <th>Turnarou nd time</th> </tr> </thead> <tbody> <tr> <td>T1</td> <td>0</td> <td>4</td> <td>VM1: 0+4=4 VM2:0 VM3:0</td> <td>VM1 : 0 VM2 : 0 VM3 : 0</td> <td>VM 1</td> <td>VM1:T1 VM2:- VM3:-</td> <td>VM1:4-0=4 VM2:0 VM3:0</td> </tr> <tr> <td>T2</td> <td>1</td> <td>5</td> <td>VM1:4 VM2:5 +1=6 Vm3:0</td> <td>VM1 : 4-1=3 VM2 : 0 VM3 : 0</td> <td>VM 2</td> <td>VM1:T1 VM2:T2 VM3:-</td> <td>VM1:4-0=4 VM2:6-1=5 VM3:0</td> </tr> <tr> <td>T3</td> <td>2</td> <td>7</td> <td>VM1 : 4 VM2 : 6 VM3 : 2+7=9</td> <td>VM1 : 4-2=2 VM2 : 6-2=4 VM3 : 0</td> <td>VM 3</td> <td>VM1:T1 VM2:T2 VM3:T3</td> <td>VM1:4-0=4 VM2:6-1=5 VM3=9-2=7</td> </tr> <tr> <td>T4</td> <td>3</td> <td>9</td> <td>VM1 : 4 VM2 :</td> <td>VM1 : 4-3=1 VM2 : 6-</td> <td>VM 1</td> <td>VM1:T1, T4 VM2:T2</td> <td>VM1:4-0=4 VM2:6-</td> </tr> </tbody> </table>	Tas k Id	Arri val time	Exec ution time	Comple tion time	Waiting time each on VMs	Assi gne d VM	Waiting queue on each VM	Turnarou nd time	T1	0	4	VM1: 0+4=4 VM2:0 VM3:0	VM1 : 0 VM2 : 0 VM3 : 0	VM 1	VM1:T1 VM2:- VM3:-	VM1:4-0=4 VM2:0 VM3:0	T2	1	5	VM1:4 VM2:5 +1=6 Vm3:0	VM1 : 4-1=3 VM2 : 0 VM3 : 0	VM 2	VM1:T1 VM2:T2 VM3:-	VM1:4-0=4 VM2:6-1=5 VM3:0	T3	2	7	VM1 : 4 VM2 : 6 VM3 : 2+7=9	VM1 : 4-2=2 VM2 : 6-2=4 VM3 : 0	VM 3	VM1:T1 VM2:T2 VM3:T3	VM1:4-0=4 VM2:6-1=5 VM3=9-2=7	T4	3	9	VM1 : 4 VM2 :	VM1 : 4-3=1 VM2 : 6-	VM 1	VM1:T1, T4 VM2:T2	VM1:4-0=4 VM2:6-
Tas k Id	Arri val time	Exec ution time	Comple tion time	Waiting time each on VMs	Assi gne d VM	Waiting queue on each VM	Turnarou nd time																																			
T1	0	4	VM1: 0+4=4 VM2:0 VM3:0	VM1 : 0 VM2 : 0 VM3 : 0	VM 1	VM1:T1 VM2:- VM3:-	VM1:4-0=4 VM2:0 VM3:0																																			
T2	1	5	VM1:4 VM2:5 +1=6 Vm3:0	VM1 : 4-1=3 VM2 : 0 VM3 : 0	VM 2	VM1:T1 VM2:T2 VM3:-	VM1:4-0=4 VM2:6-1=5 VM3:0																																			
T3	2	7	VM1 : 4 VM2 : 6 VM3 : 2+7=9	VM1 : 4-2=2 VM2 : 6-2=4 VM3 : 0	VM 3	VM1:T1 VM2:T2 VM3:T3	VM1:4-0=4 VM2:6-1=5 VM3=9-2=7																																			
T4	3	9	VM1 : 4 VM2 :	VM1 : 4-3=1 VM2 : 6-	VM 1	VM1:T1, T4 VM2:T2	VM1:4-0=4 VM2:6-																																			

				6 VM3 : 9	3=3 VM3 : 9- 3=6		VM3:T3	1=5 VM3=9- 2=7	
	T5	4	2	VM1 : 4+9=13 VM2 : 6 VM3 : 9	VM1 : 13-4=9 VM2 : 6- 4=2 VM3 : 9- 4=5	VM 2	VM1:T4 VM2:T2, T5 VM3:T3	VM1:13- 0=13 VM2:6- 1=5 VM3=9- 2=7	
	T6	5	3	VM1 : 13 VM2 : 6+2=8 VM3 : 9	VM1 : 13-5=8 VM2 : 8- 5=3 VM3 : 9- 5=4	VM 2	VM1:T4 VM2:T2, T5,T6 VM3:T3	VM1:13- 0=13 VM2:8- 1=7 VM3=9- 2=7	
	T7	6	10	VM1 : 13 VM2 : 8+3=11 VM3 : 9	VM1 : 13-6=7 VM2 : 11-6=5 VM3 : 9- 6=3	VM 3	VM1:T4 VM2:T5, T6 VM3:T3, T7	VM1:13- 0=13 VM2:11- 1=10 VM3=9- 2=7	
	T8	7	6	VM1 : 13 VM2 : 11 VM3 : 9+10=1 9	VM1 : 13-7=6 VM2 : 11-7=4 VM3 : 19-7=12	VM 2	VM1:T4 VM2:T5, T6,T8 VM3:T3, T7	VM1:13- 0=13 VM2:11- 1=10 VM3=19 -2=17	
	T9	8	2	VM1 : 13 VM2 : 11+6=1 7 VM3 : 19	VM1 : 13-8=5 VM2 : 17-8=9 VM3 : 19-8=11	VM 1	VM1:T4, T9 VM2:T6, T8 VM3:T3, T7	VM1:13- 0=13 VM2:17- 1=16 VM3=19 -2=17	
	T10	9	8	VM1 : 13+2=1 5 VM2 : 17 VM3 : 19	VM1 : 15-9=6 VM2 : 17-9=8 VM3 : 19-9=10	VM 1	VM1:T4, T9,T10 VM2:T6, T8 VM3:T7	VM1:15- 0=15 VM2:17- 1=16 VM3=19 -2=17	
	Average waiting time on VM1:(1+5+6)/15=.8 , Average waiting time on VM2:(2+3+4)/16=.56 Average waiting time on VM3:3/17=.18								
SJF	ShortestJobFirst(I)// There are I jobs While (I ≠ 0) do Accept the shortest possible job j from all I jobs. Delete j, and intervals which intersect j from I. Assign j to available VM until completion. I = I -1 End while			Assume the same cloud computing environment in the previous example  Table2.3: Task time calculation on various VMs using SJF					
	ask Id	Arri val time	Exe cuti on time	Comple tion time	Waiting time each on VMs	Assig ned VM	Waiting queue on each VM	Turnarou nd time	
	T1	0	4	VM1: 0+4=4 VM2:0 VM3:0	VM1 : 0 VM2 : 0 VM3 : 0	VM1	VM1:T1 VM2:- VM3:-	VM1:4- 0=4 VM2:0 VM3:0	
	T2	1	5	VM1:4 VM2:5 +1=6 Vm3:0	VM1 : 0 VM2 : 0 VM3 : 0	VM2	VM1:T1 VM2:T2 VM3:-	VM1:4- 0=4 VM2:6- 1=5 VM3:0	

		<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">T3</td> <td style="text-align: center;">2</td> <td style="text-align: center;">7</td> <td>VM1 : 4 VM2 : 6 VM3 : 2+7=9</td> <td>VM1 : 0 VM2 : 0 VM3 : 0</td> <td style="text-align: center;">VM3</td> <td>VM1:T1 VM2:T2 VM3:T3</td> <td>VM1:4-0=4 VM2:6-1=5 VM3=9-2=7</td> </tr> <tr> <td style="text-align: center;">T4, T5, T6</td> <td style="text-align: center;">4</td> <td style="text-align: center;">9,2, 3</td> <td>VM1 : 4 VM2 : 6 VM3 : 9</td> <td>VM1 : 0 VM2 : 6-4=2 VM3 : 9-4=5</td> <td style="text-align: center;">T5:V M1 T6:V M2 T4:V M3</td> <td>VM1:T5 VM2:T2, T6 VM3:T3, T4</td> <td>VM1:6-0=6 VM2:9-1=8 VM3=18-2=16</td> </tr> <tr> <td style="text-align: center;">T7, T8, T9, T10</td> <td style="text-align: center;">6</td> <td style="text-align: center;">10,6, 2,8</td> <td>VM1 : 6 VM2 : 6+3=9 VM3 : 9+9=18</td> <td>VM1 : 0 VM2 : 9-6=3 VM3 : 18-6=12</td> <td style="text-align: center;">T9:V M1 T8:V M1 T10: VM2 T7:V M2</td> <td>VM1:T9, T8 VM2:T10, T7 VM3:-</td> <td>VM1:14-0=14 VM2:27-1=26 VM3=18-2=16</td> </tr> </table> <p>Average waiting time on VM1:2/14=.14 Average waiting time on VM2:(2+3+8)/26=.5 Average waiting time on VM3:(5/17)=.29</p>	T3	2	7	VM1 : 4 VM2 : 6 VM3 : 2+7=9	VM1 : 0 VM2 : 0 VM3 : 0	VM3	VM1:T1 VM2:T2 VM3:T3	VM1:4-0=4 VM2:6-1=5 VM3=9-2=7	T4, T5, T6	4	9,2, 3	VM1 : 4 VM2 : 6 VM3 : 9	VM1 : 0 VM2 : 6-4=2 VM3 : 9-4=5	T5:V M1 T6:V M2 T4:V M3	VM1:T5 VM2:T2, T6 VM3:T3, T4	VM1:6-0=6 VM2:9-1=8 VM3=18-2=16	T7, T8, T9, T10	6	10,6, 2,8	VM1 : 6 VM2 : 6+3=9 VM3 : 9+9=18	VM1 : 0 VM2 : 9-6=3 VM3 : 18-6=12	T9:V M1 T8:V M1 T10: VM2 T7:V M2	VM1:T9, T8 VM2:T10, T7 VM3:-	VM1:14-0=14 VM2:27-1=26 VM3=18-2=16
T3	2	7	VM1 : 4 VM2 : 6 VM3 : 2+7=9	VM1 : 0 VM2 : 0 VM3 : 0	VM3	VM1:T1 VM2:T2 VM3:T3	VM1:4-0=4 VM2:6-1=5 VM3=9-2=7																			
T4, T5, T6	4	9,2, 3	VM1 : 4 VM2 : 6 VM3 : 9	VM1 : 0 VM2 : 6-4=2 VM3 : 9-4=5	T5:V M1 T6:V M2 T4:V M3	VM1:T5 VM2:T2, T6 VM3:T3, T4	VM1:6-0=6 VM2:9-1=8 VM3=18-2=16																			
T7, T8, T9, T10	6	10,6, 2,8	VM1 : 6 VM2 : 6+3=9 VM3 : 9+9=18	VM1 : 0 VM2 : 9-6=3 VM3 : 18-6=12	T9:V M1 T8:V M1 T10: VM2 T7:V M2	VM1:T9, T8 VM2:T10, T7 VM3:-	VM1:14-0=14 VM2:27-1=26 VM3=18-2=16																			
RR	<p>Input: Cloudletlist(tasks),VML: The list of available VMs Output: Map each cloudlet to a VM.</p> <p>Steps: NoCL:cloudletlist.size(); NoVM:VML.size(); Index:0; For j=0 to NoCL do CL:cloudletlist.get(j); Index:(index+1)mod NoVM; V:VML.get(index); Stagein:transfertime(CL,V,in); Stageout:transfertime(CL,V,out); Exec:executetime(CL,V); If(CL.AT+stagein+exec+stageout+V.RT&lt;=CL.DL)then Sendjob(CL,V) Update(V); Else Drop(CL); FailedJobs; end</p>	<p>Assume a system consists of 4 tasks ,1 VM, quantum time=100, and values corresponding to the vector (Process id , Arrival time , Execute time) are : {(T0,0,150), (T1,50,200), (T2,230,60), (T3,280,100)} then RR will be executed as follows :</p> <p style="text-align: center;">Table2.4: RR tasks scheduling</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">Time</td> <td style="text-align: center;">0</td> <td style="text-align: center;">50</td> <td style="text-align: center;">100</td> <td style="text-align: center;">200</td> <td style="text-align: center;">230</td> <td style="text-align: center;">250</td> <td style="text-align: center;">280</td> <td style="text-align: center;">350</td> <td style="text-align: center;">410</td> <td style="text-align: center;">510</td> </tr> <tr> <td style="text-align: center;">Process</td> <td style="text-align: center;">T0</td> <td style="text-align: center;">T0, T1</td> <td style="text-align: center;">T1, T0</td> <td style="text-align: center;">T0, T1</td> <td style="text-align: center;">T0, T1,</td> <td style="text-align: center;">T1, T3</td> <td style="text-align: center;">T1, T2,</td> <td style="text-align: center;">T2, T3</td> <td style="text-align: center;">T3</td> <td style="text-align: center;">-</td> </tr> </table> <p>For a cloud environment with several virtual machines the distribution of tasks on VMs is implemented in a round fashion as shown in Figure6</p> <div style="text-align: center;">  <p>Figure6:RR task scheduling</p> </div>	Time	0	50	100	200	230	250	280	350	410	510	Process	T0	T0, T1	T1, T0	T0, T1	T0, T1,	T1, T3	T1, T2,	T2, T3	T3	-		
Time	0	50	100	200	230	250	280	350	410	510																
Process	T0	T0, T1	T1, T0	T0, T1	T0, T1,	T1, T3	T1, T2,	T2, T3	T3	-																
GA	<p>Standard Genetic Algorithm (SGA)</p> <ul style="list-style-type: none"> <li>□ Produce an initial population by randomly generated individuals.</li> <li>□ Evaluate the fitness of all individuals.</li> <li>□ while termination condition not met do: <ul style="list-style-type: none"> <li>o select fitter individuals for reproduction</li> <li>o crossover between individuals</li> </ul> </li> </ul>	<p>The standard genetic algorithm (SGA) could be exploited to solve the problem of task scheduling in cloud. The steps for this are as follows:</p> <ol style="list-style-type: none"> <li>1- Initialize the population The population is generated randomly using binary encoding where each chromosome corresponds to a VM and the genes of this chromosome represents the scheduled tasks on this VM.Example on this VM1[ T4,T2,T5],VM2[T1,T7,T6],VM3[T3,T8,T9]</li> <li>2- The fitness function. The purpose related to scheduling tasks on VMs is to find the best assignment of tasks on VMj such that the completion time for tasks on VMs is minimized. This can be formulated using the following equation.</li> </ol>																								

	<ul style="list-style-type: none"> <li>o mutate individuals</li> <li>o evaluate the fitness of the modified individuals</li> <li>o Generate a new population</li> <li>□ End while</li> </ul>	<p><math>P_j = \sum P_{ij}</math> where  <math>P_j</math> is the execution time of all tasks (1...n) on VMj.  <math>P_{ij}</math> is the execution time of task <math>P_i</math> on VMj.  <math>P_{ij} = C_i/P_{sj}</math> where <math>C_i</math> is the computation complexity of tasks <math>P_i</math> and <math>P_{sj}</math> is the processing speed of VMj.</p> <ol style="list-style-type: none"> <li>3- Selection process: in this step the two random individuals are selected to do the GA operations on them in order to generate new population and the nonelected individuals are kept untouched.</li> <li>4- Crossover :is the process of exploring the search space and generating new solutions( descendents) form the original solutions(parents) by interchanging the genes of the selected chromosomes.</li> <li>5- Mutation: is to do operations such as swap,move,or replacement on gene values.</li> <li>6-Evaluation:the solutions are evaluated based on the fitness functions and the ones that achieved good fitness are chosen in the next iteration</li> </ol>
	<ol style="list-style-type: none"> <li>1- Set particle as equal to the size of ready tasks in <math>\{t_i\} \in T</math></li> <li>2- Initialize particles position randomly from VM= 1, ..., j and velocity <math>v_i</math> randomly.</li> <li>3- For each particle, calculate its fitness value.</li> <li>4- If the fitness value is better than the previous best pbest, set the current fitness value as the new pbest.</li> <li>5- After Steps 3 and 4 for all particles, select the best particle as gbest.</li> <li>6- For all particles, calculate velocity and update their positions.</li> </ol> <p>If the stopping criteria or maximum iteration is not satisfied, repeat from Step3.</p>	<p>Figure 7 shows the behavior of PSO algorithm in exploring the search space ;the particles represent the tasks which are initiated randomly (red circles) and each one will have a fitness value which will be evaluated using a fitness function at each iteration .if the fitness value is better than the previous one the local best or personal best (blue circles) are updated .The particles memorize the best positions they have achieved .The best one of all of these local best solutions represents the global best(green circle) which is in this scenario the best mapping of a task onto VM.</p>  <p style="text-align: center;">Figure7:PSO example</p>

3.1 METHODOLOGY

Modified round robin does a simple improvement on the standard round robin algorithm by dynamically taking into consideration the burst time for each incoming task entering the ready queue. The time slicing process is based on computing the average burst time (expected needed execution time for each task) of all the remaining waiting requests in the ready queue [40]. For achieving this purpose, two registers are used: SReg for storing the total burst time of the all requests in the ready queue and AReg for storing the average

burst time by dividing the value of SReg by the number of tasks residing in the ready queue.

Initially, the first job is allocated to virtual machine and takes all its burst time. Then, the scheduler begins computing the time slice for each incoming request. Each task, when it is allocated to run on a virtual machine, it will run for a time period equals to the time slice granted to it by modified round robin when entered the queue. When the time slice elapses, the task either joins the ready queue again standing at the back of the queue, or it is removed from the ready queue. Consequently, the scheduler adjusts the values of the registers by subtracting the burst time of the

Table3: Modified Round Robin Algorithm

Input: SReg (sum register), AReg(average register) , Tn (task n), BT(T)(the burst time of a task) , TQ (time quantum), Ready Queue(a stack structure for tasks waiting their turn for execution in CPU).	
Output: all tasks finish execution and leave the ready queue.	
1. Begin	
2. New request T arrives. T Enters ready queue.	
3. Update SReg and AReg Request . //SReg includes the summation of the burst time for the whole tasks residing in the ready queue and AReg is the value of SReg divided by the number of tasks exists in the queue at certain instant of time .	O(2)
4. T is loaded from ready queue into VM queue to be executed.	O(1)
5. While (Ready Queue!= NULL) do	O(N)....N=#Tasks
6. Ready Queue T.	
7. Update SReg& AReg .	O(2)
8. Load T // For Execution	O(1)
end while	
9. If (Ready Queue = NULL) then	O(1)
10. TQ =BT (T)	O(1)
11. Update SReg & AReg	O(2)
else	
12. TQ = AVG (BT of all request in Ready Queue).	O(1)
13. Update SReg & AReg	O(2)
14. // VM executes T by TQ Time	O(TQ)
15. If (T terminated) then	O(1)
16. Update SReg & AReg	O(2)
else	
17. Return T // To the Ready Queue with its updated .	O(1)
18. Burst Time (BT) .	
19. Update SReg & AReg	O(2)
20. end if	
Run time cost = 2 + 1 + n*(2+1) + 1 + [ ( prop of statements 10 and 11 say 1/2)*(1+2) + ( prop of statements 12 , 13,and 14 say 1/2)*(1+2+TQ) + 1 + [(prop of statement16 say 1/2)*2+(prop of statement17,18 say 1/2)*2 ] .	
Run time cost = 2+1+3n+1+(.5*3)+(.5*3)+(.5*QT)+1+(.5*2)+(.5*2)=10+3n+.5*QT=O(n)	

removed task from SReg and adding the value of the new joined tasks to AReg. The pseudo code for modified Round Robin Algorithm is shown in Table3 [41].

#### 4. Cloudsim Toolkit

CloudSim is a priceless open source Java toolkit developed originally in GRIDS distributed system laboratory in the University of Melbourne, Australia [42]. CloudSim is used to simulate the cloud computing system components and services in order to evaluate resource provisioning policies, experiment different CC deployments, generate a mix of workload request distributions and test the performance of various CC configurations and scenarios. These functionalities exposed by CloudSim permits the CC developer to efficiently tackle and manipulate several critical issues related to cloud computing and finally develop the best practice.

Cloudsim framework is used to model the complicated real-world CC environment and simulate its behavior. Using this tool a model consists of different components such as datacenters, host, service brokers, scheduling and allocation policies CC be generated [43].

For a CC system to start execution there should be at least one datacenter which is registered with the Cloud Information Service registry (CIS). Multiple hosts are created within each data center. The data center broker is responsible on receiving the submitted lists of cloudlets and of virtual machines and performs the allocation policy for assigning cloudlets to VMs [44].

Within CloudSim environment, each cloudlet is mapped to a VM and each VM is mapped to a host. According to this, there are two levels of VMs provisioning [45]:



- a) At the host level: the overall processing power is distributed on the available VMs (VM policy)
- b) At the VM level: the VMs processing power is distributed among the cloudlets (tasks) existing in its execution machine (VM Scheduling).

CloudSim software platform is implemented using a multi-layered design as illustrated in Fig 9. The lowest layer is the discrete event simulation engine (SimJava) which provides the core functionalities for the upper cloud layers like creating cloud system components (services, host, data center, broker, VMs), communication between components, managing simulation clock, and queuing and processing of events[46].

The next layer implemented above SimJava layer is the CloudSim simulation layer which supports fundamental issues related to management of large scale cloud infrastructure such as memory, storage, bandwidth and interfaces of VMs. It also has the responsibility on provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state. In addition, different tasks-machines allocation policies were experimented.

The highest layer in the CloudSim stack is the user code where the basic cloud entities are exposed such as number of machines in hosts and their specification, number of tasks and their requirements in the application, the number of users and their application types, and broker scheduling policies.

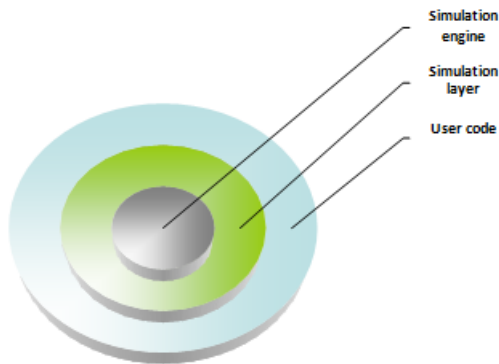


Figure 9: CloudSim Architecture

## 5. EXPERIMENT RESULTS AND ANALYSIS

The CloudSim provides cloud environment, File size, MIPS, BW, RAM, VMs configuration, Host and Data center, same as components in physical

Allocation).

cloud environments. Also, CloudSim provides scheduling environment with customization scheduling depending on user requirements. This simulation can be used to test the efficiency and performance of a schedule algorithm. It covers the need to experiment the proposed MRR scheduler and compare it with RR scheduler. Also, it is used to compare previous related work in RR algorithm and RR algorithm in our CloudSim environment in different machine specification.

Two algorithms was implemented using CloudSim simulation. In the experiments, we employed two data centers contain VMs. We used different number of VMs up to six and different number of cloudlets (tasks) up to 300. In the two algorithms, we measured the performance depending on the execution time for scheduling cloudlets on VMs.

Fig 10 and Table 4 show the results of running RR schedule on the machine with specifications:4096MB RAM, Intel® core™ i5 CPU@1.80GHz , and windows 64-bit. Then, we compared with Hicham's results of running RR schedule [46]. This is to verify the current implementation of the RR with a previous one. The figure shows that the same schedule with same parameters gets different results depending on the environment that it runs on. The difference in resource capabilities such as the processing power (number of instructions per seconds or MIPS) and RAM storage may influence the quantum time calculation process and so the performance of RR.

We run RR and MRR in the same environment using CloudSim simulator. We used the same parameters for both schedules, the number of cloudlets from 10 up to 300 tasks on a set of one VM up to six VMs. The values show a remarkable growth in the average waiting time when the number of cloudlets increases; this is because the total requested power of cloudlets increases while the available resource power is limited. The comparison is based on average waiting time needed for each schedule. Figure11 shows the average waiting time when MRR was used over different number of VMs for different Cloudlet amounts.

Figure 12 represents sample of the test for 100 cloudlets run on different numbers of VMs in CloudSim simulator. The figure shows if we use more number of VMs with the same number of Cloudlets the average waiting time decreases. The

same conclusion was inferred with 150, 200, and 300 cloudlets

Table 4: Average waiting time (AWT) in seconds for RRc on Current Machine and RRp Previous Machine

No. Cloudlets	AWT for RRp	AWT for RRc	Relative error (  RRp-RRc  /RRp)
3	5	4.33	0.15
4	5	8.5	0.41
5	10	9.6	0.04
6	13	13.33	0.02
7	18	17.28	0.04
8	22	21.5	0.02
9	32	22.22	0.44
10	33	26.2	0.26

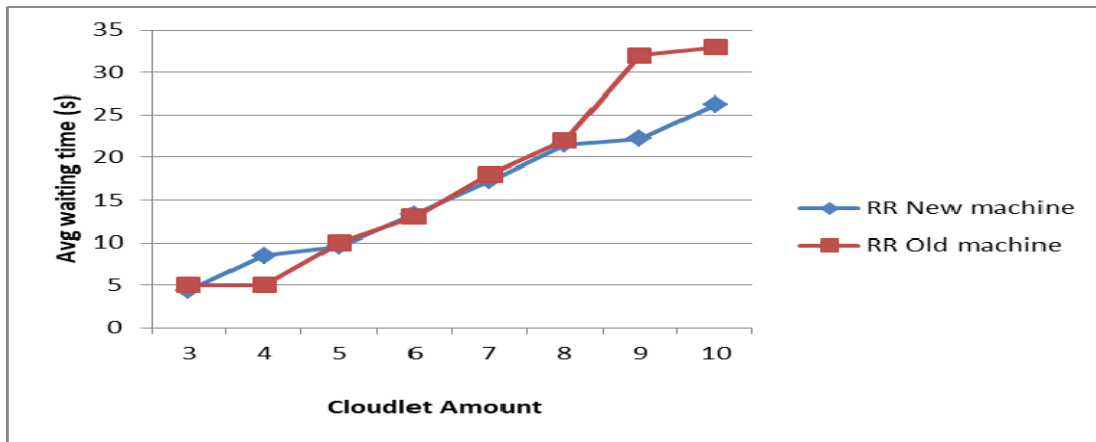


Figure 10: RR results on current machine and old machine

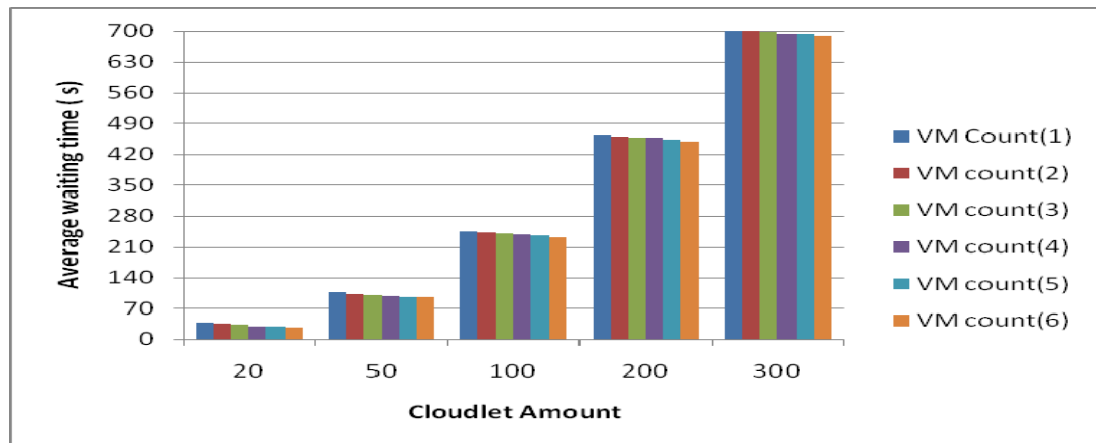


Figure 11: Average waiting time when MRR was used over different number of VMs for different Cloudlet amounts

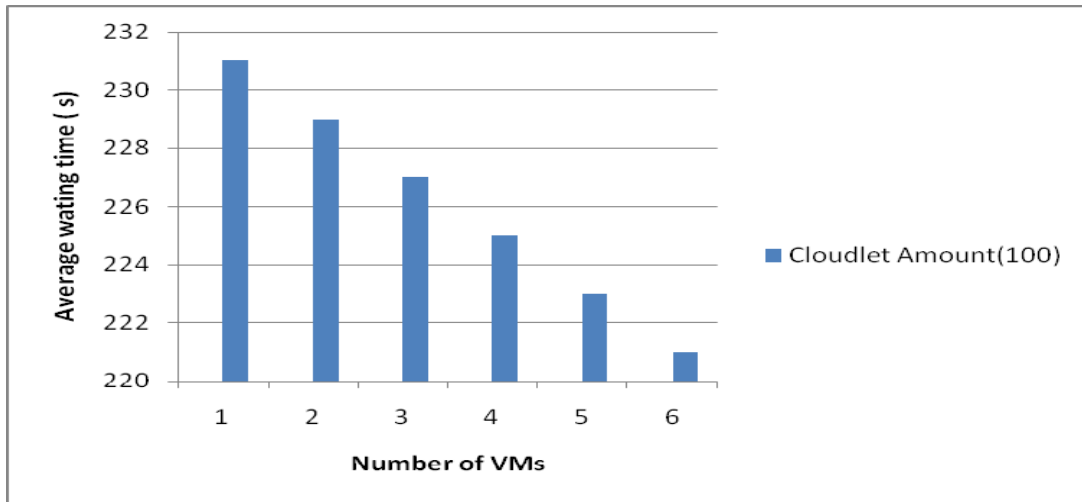


Figure 12: Average waiting time for 100 cloudlets run over different number of VMs using MRR

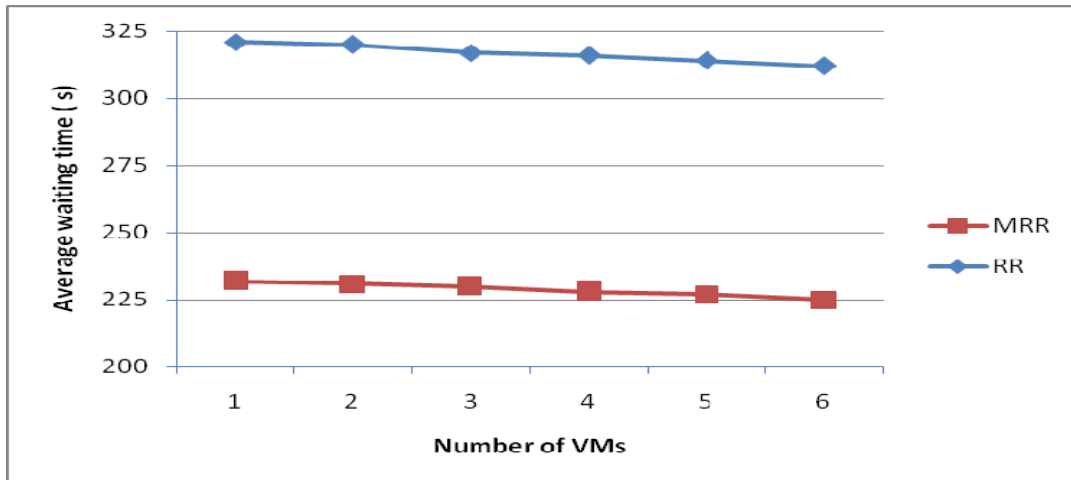


Figure 13: Average waiting time versus number of VMs for 100 cloudlets run using MRR and RR

Figure 13 shows part of the execution in CloudSim using 100 cloudlets on different number of VMs for both schedules RR and MRR. The graph shows that MRR is more efficient with less average waiting time than RR schedule with the same specifications for both algorithms.

algorithm after implementing the experiments in CloudSim. The experiments were conducted over various numbers of cloudlets 10,20,30, ..., 300 and the number of virtual machines equal two. The results show that the experimental running time and the theoretical running time have similar behavior.

Figure 14 shows the experimental complexity versus the theoretical complexity  $O(n)$  for MRR

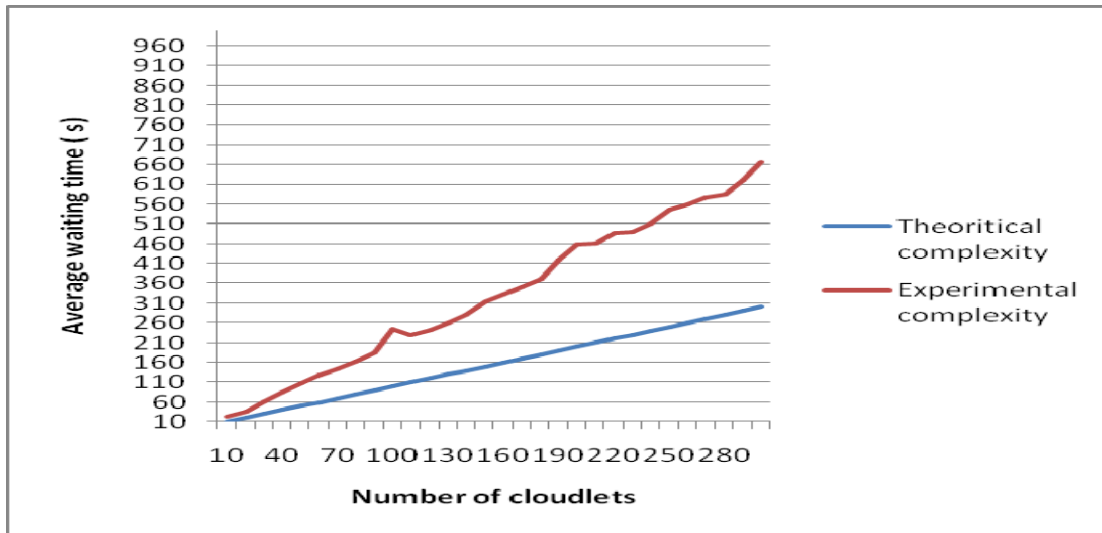


Figure 14: Comparison of experimental and theoretical MRR complexity

Table5: RR & MRR (waiting time(s))

MRR	RR	No. Cloudlets	MRR	RR	No. Cloudlets
333	437	160	23	120	10
350	449	170	36	132	20
370	479	180	62	160	30
420	526	190	85	178	40
458	552	200	108	217	50
460	568	210	128	233	60
486	587	220	144	251	70
490	592	230	162	264	80
510	615	240	184	287	90
547	548	250	245	342	100
560	663	260	230	331	110
578	674	270	241	349	120
584	691	280	260	363	130
620	723	290	281	386	140
667	768	300	315	408	150
9937	12893	Sum			

The average waiting time (AWT) using MRR is less than the AWT when using RR as shown in Table 5. For this experimented data set, the MRR is faster than RR in average waiting time for the selected quantum by up 5 times.

## 6. CONCLUSION AND FUTURE WORK

Cloud computing is a distributed based computer paradigm which is used by users to get good quality with less cost. As task scheduling is a challenge in cloud computing, different algorithms have been suggested and applied to get better results regarding utilization of system resources, response time and satisfaction of user demands. In this study, we highlight some of the task scheduling algorithms used in cloud supported by examples, namely Round Robin (RR), MaxMin, MinMin, FCFS, MCT, PSO, and GA.

Also, we studied the behavior of modified round robin task scheduling within the environment of CloudSim and compared between the performance of the RR and MRR in terms of average waiting time. The results show that when using MRR to scheduling number of Cloudlets over number of VMs, the average waiting time becomes less than when using RR, using the same numbers of cloudlets and the CC environments.

For future work, it is worth to investigate the effect of other parameters such as VMs, datacenters, memory, bandwidth for network and storage in cloud environments, and reflect that in real physical environment.

## REFERENCES:

- [1] Buyya, R., Broberg, J., & Goscinski, A. M. (Eds.). (2010). *Cloud computing: 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 Engineering and Management Research (IJEMR)*, 7(2), 479-485.
- [3] Sharieh, A., & Al-Thwaib, E. (2017, May). A mathematical model for hybrid-multi-cloud environment. In *Information Technology (ICIT), 2017 8th International Conference on* (pp. 193-199). IEEE.
- [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).
- [6] Gabriel Herrera. 19 September 2014 SaaS 101: IaaS, PaaS, SaaS and Cloud Computing. <https://starthq.com/blog/saas-101-iaas-paas-saas-and-cloud-computing>. (2018, february 18).
- [7] Vijay, P., & Anju, B. G. (2014). *An Efficient Workflow Scheduling Approach in Cloud Computing* (Doctoral dissertation). [8] Grandinetti, L. (Ed.). (2013). *Pervasive Cloud Computing Technologies: Future Outlooks and Interdisciplinary Perspectives: Future Outlooks and Interdisciplinary Perspectives*. IGI Global.
- [8] Grandinetti, L. (Ed.). (2013). *Pervasive Cloud Computing Technologies: Future Outlooks and Interdisciplinary Perspectives: Future Outlooks and Interdisciplinary Perspectives*. IGI Global.
- [9] Al-Shaikh, A., Khattab, H., Sharieh, A., & Sleit, A. (2016). Resource Utilization in Cloud Computing as an Optimization Problem. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 7(6), 336-342.
- [10] Sharieh, A., & Albdour, L. (2017). A Heuristic Approach for Service Allocation in Cloud Computing. *International Journal of Cloud Applications and Computing (IJCAC)*, 7(4), 60-74.
- [11] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50-58.
- [12] Islam, M. S. U., & Rana, B. (2017). Task Scheduling in Cloud Computing. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3(4), 642-646.
- [13] Van den Bossche, R., Vanmechelen, K., & Broeckhove, J. (2010, July). Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on* (pp. 228-235). IEEE.
- [14] Sindhu, S., & Mukherjee, S. (2011). Efficient task scheduling algorithms for cloud computing environment. In *High Performance Architecture and Grid Computing* (pp. 79-83). Springer, Berlin, Heidelberg.
- [15] Tawfeek, M. A., El-Sisi, A., Keshk, A. E., & Torkey, F. A. (2013, November). Cloud task scheduling based on ant colony optimization. In *Computer Engineering & Systems (ICCES), 2013 8th International*.
- [16] Gogulan, R., Kavitha, A., & Kumar, U. K. (2012). A multiple pheromone algorithm for cloud scheduling with various QOS requirements. *Int. J. Comput. Sci*, 9, 3.
- [17] Kaleeswaran, A., Ramasamy, V., & Vivekanandan, P. (2013). Dynamic scheduling of data using genetic algorithm in cloud computing. *Park College of Engineering and Technology, Coimbatore, India*.
- [18] Agarwal, D., & Jain, S. (2014). Efficient optimal algorithm of task scheduling in cloud computing environment. *arXiv preprint arXiv:1404.2076*.
- [19] Lakra, A. V., & Yadav, D. K. (2015). Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. *Procedia Computer Science*, 48, 107-113.
- [20] Branch, K. (2015). A novel task scheduling method in cloud environment using cuckoo optimization algorithm. *International Journal of Cloud-Computing and Super-Computing*, 2(2), 7-20.
- [21] Hamad, S. A., & Omara, F. A. (2016). Genetic-based task scheduling algorithm in cloud computing environment. *International Journal of Advanced computer Science and Applications*, 7(4), 550-556.
- [22] Dandhwani, Vanita, and Vipul Vekariya. "Evolutionary Algorithm Using K-mean For Task Scheduling in Cloud Computing."
- [23] Madni, S. H. H., Latiff, M. S. A., Abdullahi, M., & Usman, M. J. (2017). Performance comparison of heuristic algorithms for task



- scheduling in IaaS cloud computing environment. *PloS one*, 12(5), e0176321.
- [24] Samal, P., & Mishra, P. (2013). Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing. *International Journal of computer science and Information Technologies*, 4(3), 416-419.
- [25] Bhoi, U., & Ramanuj, P. N. (2013). Enhanced max-min task scheduling algorithm in cloud computing. *International Journal of Application in Engineering and Management (IJAIEEM)*, 2(4), 259-264.
- [26] Devipriya, S., & Ramesh, C. (2013, December). Improved max-min heuristic model for task scheduling in cloud. In *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on* (pp. 883-888). IEEE.
- [27] Gupta, A. K., & Rawat, P. S. (2017). Efficient Resource Utilization in Virtual Cloud Computing Environment. *International Journal of Computer Applications*, 168(11).
- [28] Salot, P. (2013). A survey of various scheduling algorithm in cloud computing environment. *International Journal of Research in Engineering and Technology*, 2(2), 131-135.
- [29] Vignesh, V., Sendhil Kumar, K. S., & Jaisankar, N. (2013). Resource management and scheduling in cloud environment. *International journal of scientific and research publications*, 3(6), 1.
- [30] Panda, S. K., & Jana, P. K. (2015). Efficient task scheduling algorithms for heterogeneous multi-cloud environment. *The Journal of Supercomputing*, 71(4), 1505-1533.
- [31] Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010, April). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on* (pp. 400-407). IEEE.
- [32] Guo, L., Zhao, S., Shen, S., & Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. *JNW*, 7(3), 547-553.
- [33] Jang, S. H., Kim, T. Y., Kim, J. K., & Lee, J. S. (2012). The study of genetic algorithm-based task scheduling for cloud computing. *International Journal of Control and Automation*, 5(4), 157-162.
- [34] Zhao, C., Zhang, S., Liu, Q., Xie, J., & Hu, J. (2009, September). Independent tasks scheduling based on genetic algorithm in cloud computing. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on* (pp. 1-4). IEEE.
- [35] Shreedhar, M., & Varghese, G. (1996). Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on networking*, 4(3), 375-385.
- [36] Yigitbasi, N., Iosup, A., Epema, D., & Ostermann, S. (2009, May). C-meter: A framework for performance analysis of computing clouds. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on* (pp. 472-477). IEEE.
- [37] Nieh, J., Vaill, C., & Zhong, H. (2001, June). Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler. In *USENIX Annual Technical Conference, General Track* (pp. 245-259).
- [38] Bernstein, E., & Vazirani, U. (1997). Quantum complexity theory. *SIAM Journal on Computing*, 26(5), 1411-1473.
- [39] Matarneh, R. J. (2009). Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes. *American Journal of Applied Sciences*, 6(10), 1831.
- [40] Pradhan, P., Behera, P. K., & Ray, B. N. B. (2016). Modified Round Robin Algorithm for resource allocation in cloud computing. *Procedia Computer Science*, 85, 878-890.
- [41] CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. (2017, December 16). <http://www.cloudbus.org/cloudsim/>.
- [42] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1), 23-50.
- [43] Buyya, R., Ranjan, R., & Calheiros, R. N. (2009, June). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on* (pp. 1-11). IEEE.
- [44] Goyal, T., Singh, A., & Agrawal, A. (2012). Cloudsim: simulator for cloud computing infrastructure and modeling. *Procedia Engineering*, 38, 3566-3572.



- [45] Tian, W., Xu, M., Chen, A., Li, G., Wang, X., & Chen, Y. (2015). Open-source simulators for cloud computing: Comparative study and challenging issues. *Simulation Modeling Practice and Theory*, 58, 239-254.
- [46] Tani, Hicham Gibet, and Chaker El Amrani. "Smarter round robin scheduling algorithm for cloud computing and big data." *Journal of Data Mining and Digital Humanities* (2018).