International Journal of Computer Applications (0975 – 8887)
Volume 174 – No. 32, April 2021

Evaluation of Different Virtual Machine Scheduling Algorithms in Cloud Computing Environment

Oladoja I.P.
Department of Computer Science, The Federal University of Technology Akure, Nigeria

Adewale O.S.
Department of Computer Science, The Federal University of Technology Akure, Nigeria

Oluwadare S.A.
Department of Computer Science, The Federal University of Technology Akure, Nigeria

Oyekanmi E.O.
Department of Mathematical Science, Achievers University Owo, Nigeria

ABSTRACT
Resource Scheduling is a complicated task in cloud computing, as required resources are limited and the number of users increase day by day. Thus, it is important to manage these resources in a way that they are properly utilized and the waiting time is reduced. Virtual machine (VM) scheduling algorithms are used to schedule the VM requests to the Physical Machines (PM) of a Data Center to fulfill the requirements of the requested resources. Herein, the performance efficiencies of four VM scheduling algorithms, namely: First-Come First-Serve (FCFS); Resource aware scheduling algorithm (RASA); Improved Max-Min algorithm; and Median-Based improved Max-Min were evaluated and compared using CloudSim. The Makespan, Resource utilization and Throughput calculations were used to determine the minimum makespan, maximum resource utilization, and throughput for each of the VM scheduling algorithms. The four VM scheduling algorithms were implemented, the optimization metrics were calculated, and the best algorithm was determined using the three optimization criteria. The study showed that the Median-Based improved Max-min algorithm had minimum makespan (14units time) and maximum resource utilization (2.1607) and throughput (0.714).

General Terms
Scheduling algorithm, cloud simulation, cloud computing

Keywords
Cloud simulation; algorithms; virtual machine scheduling; cloud computing

1. INTRODUCTION
Cloud computing has been coined as an umbrella term to describe a category of sophisticated on-demand computing services, initially offered by commercial providers, such as Amazon, Google, and Microsoft. It denotes a model on which a computing infrastructure is viewed as a "cloud," from which businesses and individuals can access applications from anywhere in the world on demand [1]. The increase in the popularity of cloud computing systems that rent computing resources on-demand, bill on a pay-as-you-go basis, and multiplex physical infrastructure for many users has been a dramatic process, occurring in both academics and private organizations.

Virtualization, is another term for abstraction in computer science, is it the creation of virtual version of a device or service, such as, a hardware platform, OS, storage device, or network resources. It is an art of slicing the IT hardware into partitions, by implementing virtualization technology or hypervisors on top of the IT hardware. This converts the physical infrastructure into virtual servers, virtual storage, and virtual networks [2]. The usual goal of virtualization is to centralize administrative tasks, while improving scalability and workloads. Virtualization allows the user to see the infrastructure of a network through a process of aggregation.

2. RELATED WORKS
A scheduling parallel job, using migration and consolidation in cloud based, on modified FCFS scheduling has been proposed [3]. This is two level scheduling that is based on foreground VM's, and background VM's. The processes in the Foreground VM's are scheduled on the basis of First Come First Serve (FCFS) scheduling, while processes in the Background VM's are scheduled on the basis of Shortest Job (SJF) First scheduling approach. All background VM's communicate with one or more foreground VM's. Any background VM whose current allocation of process is less than 96% can accommodate a new process, otherwise it will not be allowed to accommodate a new process. Migrations are only performed during scheduling, if certain processors remain idle for long periods of time. The results show that the response time can be significantly reduced by their algorithm.
A dynamic VM allocation algorithm that is based on clustering has been proposed [4]. The study showed an improved utilization of the resources, a reduction in the data center debts, and a better performance of the algorithm, when compared with load balancing.

Psychas and Ghaderi [5] worked on non-preemptive VM scheduling in the cloud. The problem of scheduling VMs in a distributed server platform in cloud computing applications was the focus of study. The VMs arrive dynamically over time to the system, and require a certain number of resources (e.g., memory, CPU, etc.) for the duration of their service. In order to avoid costly preemptions, a non-preemptive scheduling was proposed, where each VM has to be assigned to a server which has enough residual capacity to accommodate it, and once a VM is assigned to a server, its service cannot be disrupted (preempted). Prior approaches to this problem either have high complexity, require synchronization among the servers, or yield queue sizes/delays, which are excessively large. Extensive simulation results, using both synthetic and real traffic traces were presented to verify the performance of the algorithm. The evaluation results, using synthetic and real traffic traces showed that the algorithm outperformed the other methods when the number of servers or the traffic intensity scales. The study could not show how to incorporate preemptions (through proper preemption cost models), or provide deadline (strict delay) and fairness of task was not guarantee.

Guo et al., [6] worked on optimal scheduling of VMs in Queueing Cloud Computing Systems with a Heterogeneous workload. The study focused on the delay-optimal virtual machine (VM) scheduling problem in cloud computing systems, which have a constant amount of infrastructure resources, such as CPU, memory, and storage in the resource pool. A low-complexity online scheme that combines the shortest-job-first (SJF) buffering and min–min best fit (MMBF) scheduling algorithms, i.e., SJF-MMBF, was proposed to determine the solutions, while another scheme that combines the SJF buffering and reinforced learning (RL)-based scheduling algorithms, i.e., SJF-RL, was further proposed to avoid the potential of job starvation in SJF-MMBF. The simulation results showed that the SJF-RL scheme achieved its goal of delay-optimal scheduling of VMs. This is by providing low delay performance, in terms of average job completion time, and acceptable throughput performance, in terms of job hosting rate in a queueing cloud system. The workloads ranged from light-loaded to heavy-loaded, and from slightly dynamic to highly dynamic. The SJF-MMB result showed a sub-delay-optimal in a heavy-loaded and highly dynamic environment, but it is efficient in throughput performance in terms of job hosting rate provisioning. The study did not show the non-preemptive assumption, and also failed to investigate the efficiency of the method adopted. They also did not test the convergence rate of SJF-RL in environments similitude of commercial servers.

An efficient strategy of VMs scheduling, based on physicals resources and temperature thresholds, was proposed [7]. The study focused on the high increase of VM migrations, SLA violations and energy consumption; thereby proposing two scheduling algorithms of the VMs of a Data Center in cloud computing, named SchedCT and SchedCRT. These proposed approaches were based on thresholds of resources, which are the most important factor that consumes a high quantity of energy in a server. The SchedCT was based on dynamic CPU utilization and temperature thresholds, while the SchedCR also considers the CPU utilization, Ram capacity and temperature thresholds. These approaches have efficiently decreased the energy consumption of the data centers, the number of VM migrations, and SLA violations, which ultimately reduced the emission of CO2 gas. The limitations of the work are that the approach was not applied in a real cloud computing environment, and other physical resource thresholds, namely; storage capacity, bandwidth and network load were not considered. Furthermore, the approach was not applied to a model of heterogeneous data centers and migration between VMs servers.

An appraisal of the previous studies showed that the majority used different algorithms, and there was no consideration for if the VM is idle or not during the completion time of the scheduling. Since resources need to be allocated and scheduled in a way that providers can achieve high resource utilization, so that users can meet their applications' performance requirements with minimum makespan, the present study aimed at the need to maximize the number of resources, and maximize the throughput of an application. Thus, the performance efficiencies of four VM scheduling algorithms, namely: First-Come First-Serve VM; Resource aware; improved max-min algorithm; and Median-Based improved Max-Min were evaluated and compared using CloudSim.

3. VM SCHEDULING

A virtual machine (VM) is a type of computer application that is used to create a virtual environment. In other word, the software simulates another environment. The creation of this virtual environment is referred to as Virtualization. The VM allocation is a process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the Software as a service (SaaS) provider. Allocation of application-specific VMs to hosts in a Cloud-based data center is the responsibility of a VM Allocation controller component (called VmAllocationPolicy). By default, VmAllocation Policy implements a straightforward policy that allocates VMs to the Host on a First-Come-First-Serve (FCFS) basis. Hardware requirements, such as the number of processing cores, memory, and storage, form the basis for such provisioning, [8].

For each Host component, the allocation of processing cores to VMs is done based on a host allocation policy. This policy takes into account several hardware characteristics, such as number of CPU cores, CPU share, and amount of memory (physical and secondary) that are allocated to a given VM instance. Each host component also instantiates a VM scheduler component, which can either implement the space-shared or the time-shared policy for allocating cores to VMs, [9]. Hence, in order to allow simulation of different provisioning policies under varying levels, CloudSim, which supports VM provisioning at two levels was used. The first is at the host level, while the second is at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core will be assigned to each VM, while at the VM level the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine.

Herein, we consider a task unit as a better abstraction of an application service being hosted in the VM, therefore, one VM is assigned with a host of one CPU. Therefore, in total, four hosts are created, but at each level. The CloudSim implements the space-shared provisioning policies. The reason for this is that, the virtualization tools that was used in carry out this study has XEN, which supports both time-shared policy and space-shared policy, [10]. The algorithm for space shared is given as follows:

```
                $T \rightarrow Set\ of\ tasks$
Virtual machine $V \leftarrow \emptyset$ is empty for each tasks $t_i \in T$
Do Enqueue (Q, T), where Q is the Queue // step 1
Completion time of task $[t] \leftarrow C[t]$
$c[t] \leftarrow 0$
Enqueue (V, $t_i$) // step 2
$C[t] \leftarrow (v, t_i)$ // step 3
Dequeue (v)
while $Q \neq \emptyset$
Enqueue $(v, t_{i=1})$
$C[t] \leftarrow (v, t_{i+1})$
Dequeue (v)
while $Q = 0$
IF $Q = \emptyset$ //step 4
Enqueue (Q, T) // 5
End. [11]
```

4. SCHEDULING ALGORITHMS

Scheduling refers to the set of policies for managing the order of work to be executed by a computing system, while task scheduling in Cloud data center is a set of instructions and factors that determines and select the task to be executed on the available resources between a collection of possible tasks at a particular time [12]. In Cloud data center, the task scheduling algorithms are responsible for allocating the tasks submitted by the users to the available resources. Scheduling manages the CPU memory, and to achieve maximum resource utilization, it requires good scheduling policies [13]. The main advantage of task scheduling algorithm is to achieve a high-performance computing and the best system throughput.

There are many types of Scheduling, according to different policies, such as preemptive and non-preemptive scheduling, static and dynamic scheduling, Immediate and batch scheduling, centralize and distributed scheduling. VM scheduling policies like FCFS, RASA, Improved Max-Min and Median-based improved Max-Min were evaluated in this study.

4.1. First-Come First-Serve VM Scheduling Algorithm

The most intuitive and simplest technique is to allow the first process submitted to run first. In effect, processes are inserted into the tail of a queue when they are submitted. The next process is taken from the head of the queue when it finishes running. The process is allocated to the CPU, which has least burst time. A scheduler arranges the processes, with the least burst time at the head of the queue and longest burst time at the tail of the queue. The running process is then removed from the queue. The code for FCFS scheduling is simple to write and understand. It is for parallel processing, and targets the resource having the least waiting line up time and is chosen for the received job [14]. The CloudSim toolkit supports FCFS scheduling plan for interior scheduling tasks. The limitations of First come first serve is that it is non-preemptive. The shortest tasks which are based at the back of the queue must wait for long tasks at the front to complete. Once the CPU has been allocated to a process, that process keeps the CPU until that task is finished before it releases the CPU, either by terminating or by requesting I/O. The FCFS algorithm is thus particularly troublesome for time-sharing systems, where it is important that each user get a share of the CPU at regular intervals. It would be disastrous to allow one process to keep the CPU for an extended period.

4.2. Resource Aware Scheduling Algorithm (RASA)

This is a combination of both Max-min and Min-min algorithms. In RASA, the appraisal of the completion time for each task on available resources is calculated, after which the Max-min and Min-min algorithms are applied alternatively, as shown in Fig 1, thereby making use of the advantage of both algorithms and avoiding their drawbacks [15]. RASA executes small tasks to avoid delays of executing large tasks and also support simultaneous executions of large and small tasks.

```
Start
Create different classes of job, C
Assign next job to the belong class
Sort the class tasks based on Max-weight, Min-Weight of the execution time interchangeably
//R is the resources
for (i = 0; i<|C|; i++) // |C| is the total number of classes
        Assign next task in $C_i$ on min $(R_{1_{completiontime}}, \ldots, R_{|R|_{completion}}$ resource
        Remove the task from the selected class list
        Next class
end
End
```

Figure 1: Resource Aware Scheduling Algorithm

4.3. Improved Max-Min Algorithm

Improved Max-min is based on the expected execution time instead of complete time as a basis for scheduling of task. The algorithm calculates the expected completion time of the submitted tasks on each resource and then schedule the task with the overall maximum expected execution time to a resource with minimum overall completion time. Finally, the scheduled task is removed from the task list [16].

1.for all submitted tasks in meta-task; Ti
2.for all resources; Rj
3.Cij = Eij + rj
4.While meta-task is not empty
5.find task Tk costs maximum execution time.
6.assignTkto the resource Rj which gives minimum completion time.

```
7.remove Tk from meta-tasks set
8.update rj for selected Rj
9.update Cij for all
```

4.4 Median-based improved Max-Min algorithms

The algorithm for improved max-min was modified by calculating the median of the average value of the completion time of the resources in each of the datacenter used in this paper. The minimum out of the two datacenters is calculated and the corresponding resource (virtual machine) is used to process the meta-task [11]. The algorithm is written as follows:

```
1. Begin
2. Create task-list        //set of tasks in FCFS order.
3. for (i=0; i<=|VM|)//|VM| is the number of virtual machine present in each host
4. assign task to VMi using FCFS
5. remove task from task-list
6. endfor
7. Sort remaining task in ascending order of their execution time
8. Find median of the sorted tasks based on execution time
9. Assign the task that falls at the median value to the resource with minimum completion time
10. Remove the task from the task-list
11. Repeat line 7 to 10 while (task-list! = empty)
12. End
```

5. SIMULATION REPORT

Table 1 shows the output of the simulation of ten (10) cloudlets in FCFS order in a typical datacenter. The particular virtual machine that processes the task were listed under the VM ID column in Table 1. The next column to VM ID shows the execution time for each of the cloudlet in Table 1. Each respective task size is listed under column CloudletLenght.

Table1: Simulation report of 10 cloudlets in FCFS order:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	CloudletLength	Start Time	Finish Time
0	SUCCESS	2	0	7	6562	0	7
1	SUCCESS	2	1	8	8062	0	8
4	SUCCESS	2	0	5	5000	7	12
3	SUCCESS	2	3	14	14135	0	14
2	SUCCESS	2	2	23	22597	0	23
6	SUCCESS	2	2	2	1561	23	24
5	SUCCESS	2	1	19	19355	8	28
8	SUCCESS	2	0	20	19891	12	32
7	SUCCESS	2	3	18	17948	14	32
9	SUCCESS	2	1	5	5207	28	33

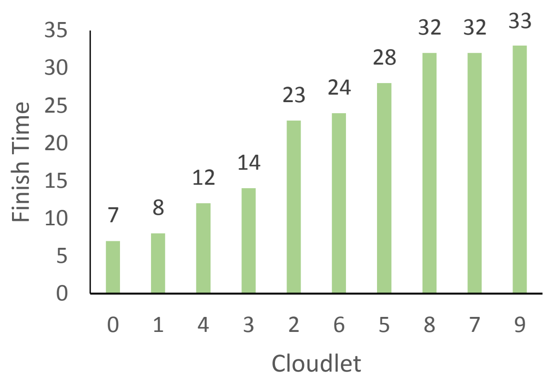

Figure 2: Graph of finishing time against cloudlets

The results presented in Fig. 3-6 shows the Gant chart of the scheduling of tasks following FCFS, RASA, Improved Max-Min and Median-Based Improved Max-min respectively.

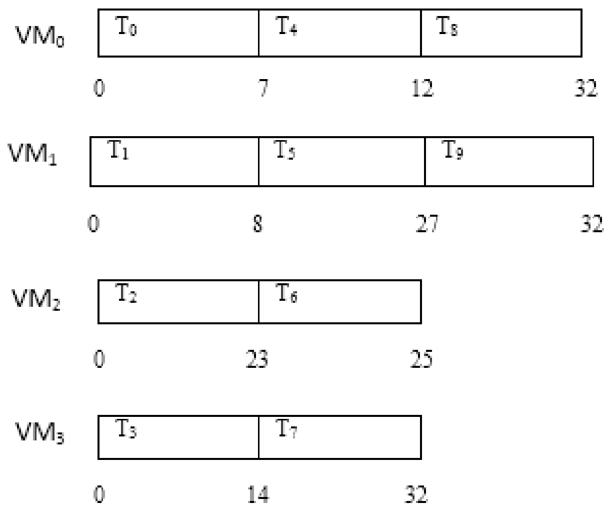

Figure 3: Gantt chart for FCFS scheduling algorithm

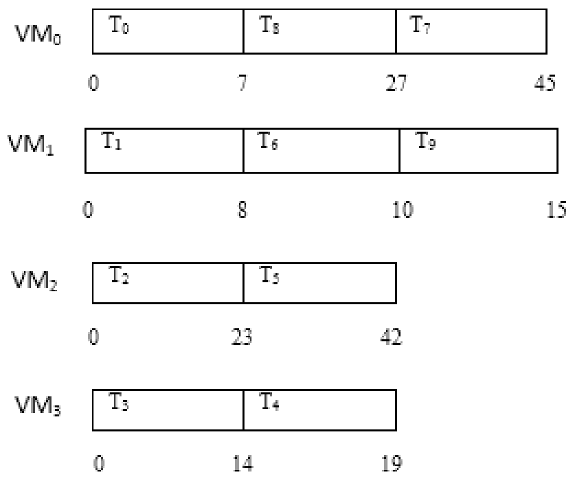

Figure 4: Gant chart for RASA scheduling algorithm

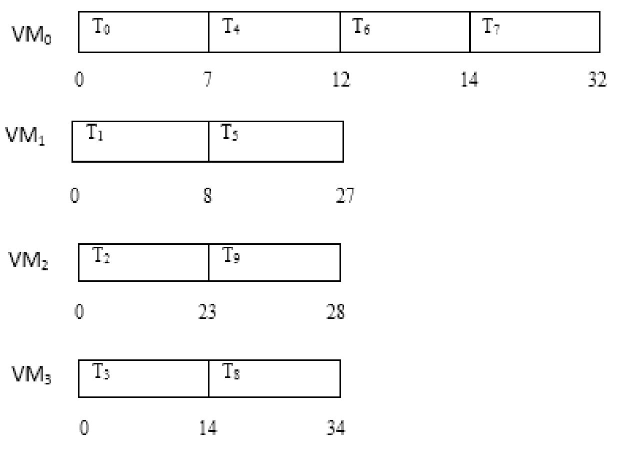

Figure 5: Gant chart for Improved max-min scheduling algorithm

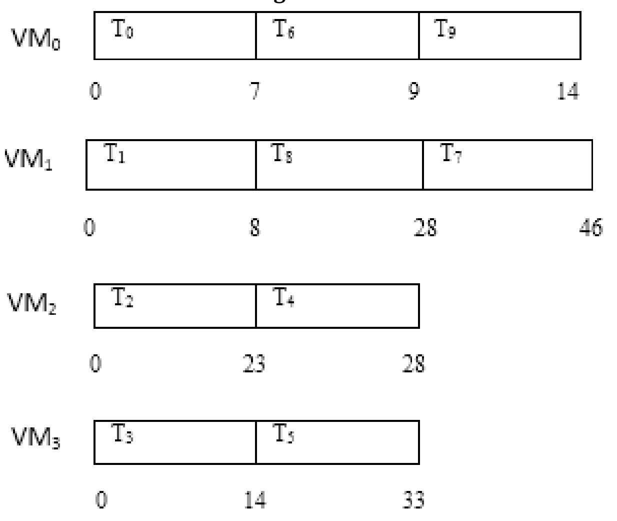

Figure 6: Gant chart for Median-Based improved max-min scheduling algorithm

6. RESULTS

Makespan, Resource utilization and Throughput are the optimization criteria that is calculated in this paper. Makespan indicates the finishing time of the last task. It is an optimization criterion that most users desire during execution of their application.

$$Makespan = max_{i \in Tasks} \{F_i\} \qquad (1)$$

For the analysis of algorithms, the Makespan values (Fig. 7) were calculated using Equation 1.

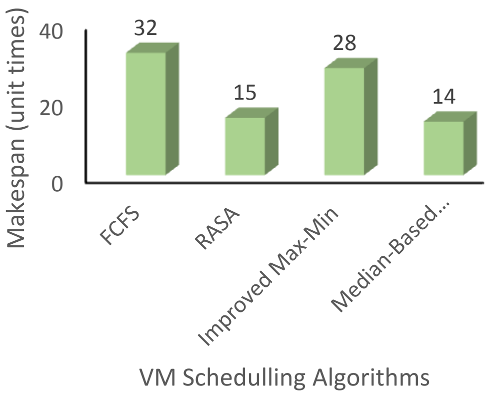

Figure 7: Makespan of the different algorithms

Resource utilization is an important criterion that depicts the maximization of resource utilization i.e., keeping resources as busy as possible. The values (Fig. 8) were derived from Equation 2:

$$Resource\ utilization = \frac{\sum_{i=1}^{n} time\ taken\ for\ resource\ i\ to\ finish\ all\ tasks}{makespan \times n} \qquad (2)$$

Where, n, is the number of resources.

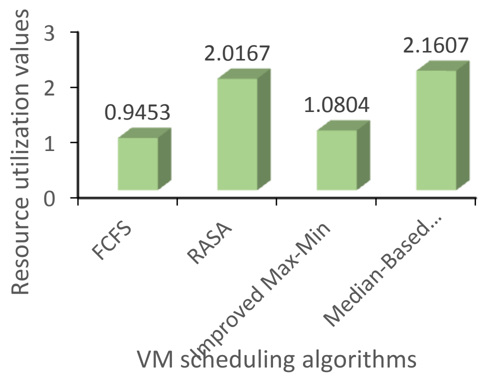

Figure 8: Resource utilization values of the different algorithms

Throughput, which is the total number of jobs executed per unit time, is a measure of how many units of task a system can process in a given amount of time. The values for throughput, derived from Equation 3, for the different algorithms are presented in Fig. 9.

$$Throughput = \frac{Total\ number\ of\ tasks}{time(seconds)} \qquad (3)$$

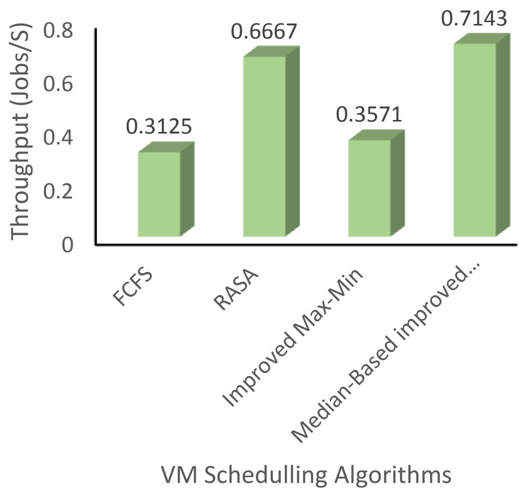

Figure 9: The Throughput values of the different algorithms

It can be deduced that the Makespan for Median-Based improved Max-Min is reduced during processing when compared to the others and it shows that the system was highly utilized when compared to improved Max-min scheduling. The median-based improved max-min had a value of 2.1607, followed by RASA in resource utilization.

The results indicate that the proposed algorithm Median-Based Improved Max-Min has maximized the resources during scheduling.

7. CONCLUSION

The optimization metrics in VM Scheduling determines how many processing cores of a host are allocated to virtual machines and how many processing cores will be delegated to each VM. The algorithms such as, First-Come First-Serve, RASA, Improved Max-Min and Median-Based Improve Max-Min are implemented in cloud computing environment using the simulation tool CloudSim. Makespan, Resource utilization and Throughput calculation is made for the above-mentioned algorithms. Median-Based Improved Max-Min algorithm has higher values for all optimization metrics calculated and hence it shows that its better among the other three algorithms.

8. REFERENCES

[1] Buyya R., Ranjan R., and Calheiros, R.N., (2009). “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities”. *In International Conference on High Performance Computing and Simulation (HPCS*).

[2] Shamir J. (2020). “5 benefits of virtualization”. Accessed on 02-02-2021, URL: https://www.ibm.com/cloud/blog/5-benefits-of-virtualization.

[3] Liu Chen, Qiu Cai & Huang “Scheduling Parallel Jobs using Migration & Consolidation in the Cloud”, Hindwai Publications of Mathematical Problems in Engineering, July 2012.

[4] Panchal and Kapoor “Dynamic VM Allocation Algorithm using Clustering in Cloud Computing”, International Journal of Advance Research in Computer Science & Software Engineering, Issue –9, Vol. 3, September 2013 [2277 –128X.

[5] Konstantinos Psychas, and Javad Ghaderi. (2017). “On Non-Preemptive VM Scheduling in the Cloud”. Proc. ACMMeas. Anal. Comput. Syst.1, 2, Article 35 (December 2017), 29 pages. https://doi.org/10.1145/31544.

[6] Mian Guo, Quansheng Guan, and Wende KE, (2018). “Optimal Scheduling of VMs in Queueing Cloud Computing Systems with a Heterogeneous Workload”. Digital Object Identifier 10.1109/ACCESS.2018.2801319.

[7] Djouhra Dad and Ghalem Belalem,(2020).”Efficient strategies of VMs scheduling based on physicals resources and temperature thresholds”. International Journal of Cloud Applications and Computing (IJCAC) 10(3).

[8] Quiroz A, kim H, Parashar M, Gnanasambandam N, and Sharma N., (2009). Towards autonomic workload provisioning for enterprise grids and clouds. *Proceedings of the 10th IEEE/ACM international conference on grid computing (grid 2009), Banf, AB, Canada,*50–57.

[9] Jinhua Hu, Jianhua Gu, Guofei Sun and Tianhai Zhao, (2010). "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment”. *Parallel architectures, algorithms and programming (PAAP), third international symposium* ,18(20),89-96.

[10] George Amalarethinam D.I. and Muthulakshmi P., (2011). “An overview of the scheduling policies and algorithms in grid computing ". *International journal of research and reviews in computer science, 2(2), 280-294.*

[11] I.P. Oladoja, O.S. Adewale, S.A. Oluwadare and E.O. Oyekanmi (2021). “A Threshold-based Tournament Resource Allocation in Cloud Computing Environment” *Asian Journal of Research in Computer Science*, Page 1-13.

[12] Er-Raji N., Benabbou F. and Eddaoui A., (2016). Task Scheduling Algorithms in the Cloud Computing Environment: Survey and Solutions. *International Journal of Advanced Research in Computer Science and Software Engineering*, 6(1), 604-608. 2277-128X.

[13] Bhavisha K. and Bhumi M., (2015). Review on Max-Min Task Scheduling Algorithm for Cloud Computing. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 2 (3),781-784. 2349-5162.

[14] Hoos H.H., and Stützle T., (2004). Stochastic Local Search: *Foundations and Applications, Elsevier*, Amsterdam, The Netherlands.

[15] Saeed P., Reza E., 2009. RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its applications, Vol. 3, p. 91-99.

[16] Elzeki O.M., Reshad M.Z. and Elsoud M.A. Improved Max-Min Algorithm in Cloud Computing.International Journal of Computer Applications, vol 50, No 12, July 2012.