HAMM: A Hybrid Algorithm of Min-Min and Max-Min Task Scheduling Algorithms in Cloud Computing

Ibrahim A. Thiyeb, Sharaf A. Alhomdy

Abstract: ***Nowadays, with the huge development of information and computing technologies, the cloud computing is becoming the highly scalable and widely computing technology used in the world that bases on pay-per-use, remotely access, Internet-based and on-demand concepts in which providing customers with a shared of configurable resources. But, with the highly incoming user's requests, the task scheduling and resource allocation are becoming major requirements for efficient and effective load balancing of a workload among cloud resources to enhance the overall cloud system performance. For these reasons, various types of task scheduling algorithms are introduced such as traditional, heuristic, and meta-heuristic. A heuristic task scheduling algorithms like MET, MCT, Min-Min, and Max-Min are playing an important role for solving the task scheduling problem. This paper proposes a new hybrid algorithm in cloud computing environment that based on two heuristic algorithms; Min-Min and Max-Min algorithms. To evaluate this algorithm, the Cloudsim simulator has been used with different optimization parameters; makespan, average of resource utilization, load balancing, average of waiting time and concurrent execution between small length tasks and long size tasks. The results show that the proposed algorithm is better than the two algorithms Min-Min and Max-Min for those parameters.***

Keywords **:** ***Task Scheduling, Load Balancing, Heuristic Algorithms, Makespan, Max-Min, Min-Min, Resource Utilization***

I. INTRODUCTION

With the advancement and increasing need of information technology, the cloud computing is coming to stay as a suitable solution for both individuals and business needs, provides to them with a large scalable and virtualized cloud resources such as hardware, servers, memory, network, storage, and services in a dynamic manner, on-demand, remote-access and pay-per-use over the world via the internet [1]. Besides that, the cloud computing has several benefits and characteristics compared to other computing technologies; it is a virtualized, cost- effective, resource pooling, independent of device and location, elastic, broad network access that available all times and accessing from anywhere via internet or private channels. It saves high expenses for creating data centers, maintenance, disaster recovery, power consumption, and technical staff. Thus, the main goal of cloud computing is to get best utilization of computing resources and distributed systems, by combining them to achieve higher throughput and to be suitable for large scale systems and organizations [2] [3] [4].

The cloud computing provides the following logical cloud service delivery types; Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In addition, there are many forms for the deployment in cloud computing. Public; in this type, cloud resources can be accessed by customers publically via an Internet connection and interfaces such as web browsers. Private clouds; a private cloud is established for a specific group or organization and limits access to just that group like intranet access in network. Hybrid; a hybrid cloud is essentially a combination of at least two clouds, where the clouds included are a mixture of public, private. Community; it is shared among two or more organizations that have similar cloud requirements [2].

There are several issues and challenges in cloud computing; cost, security, accessibility, completion time of user tasks, availability, flexibility, performance monitoring, requires a constant & speedy Internet connection, consistent and robust service abstractions, VM allocation and migration, interoperability and portability, task scheduling, scale and QoS management, and efficient load balancing. Most of researchers consider task scheduling, resource allocation and load balancing are the major concerns in the cloud computing and distribution systems, they play a significance role to enhance overall system performance [2][3] [5][6][7].

In this paper, a new hybrid scheduling algorithm has been proposed. It is named as Hybrid Algorithm of Min-Min and Max-Min (HAMM). From its name, it depends on the two traditional heuristic algorithms; Min-Min and Max-Min, to utilize from their advantages and overcome their disadvantages. In most of cases, it gives better when compared to both Min-Min and Max-Min, in the following parameters; makespan, average of utilization, average of waiting time, efficient concurrent execution between small tasks and large length tasks, and load balancing.

The rest of this paper is organized as follows. Section II and III provide the related task scheduling algorithms. Section IV presents the proposed algorithm, its flowchart and pseudo code. Simulation and analysis including Cloudsim simulator tool is defined in section V. Section VI performs result and discussion. Finally, Section VII describes the conclusion and future work.

Revised Manuscript Received on October 20, 2020.

***** Correspondence Author

Ibrahim A. Thiyeb*, Postgraduate Student IT Department, FCIT, Sana'a University, Sana'a, Yemen, E-mail: ibrahimthiib@gmail.com.

Dr. Sharaf A. Alhomdy*, Associate Prof., IT Department, FCIT, Sana'a University, Sana'a, Yemen, E-mail: sharafalhomdy@gmail.com .

II. TASK SCHEDULING

Task scheduling is the action of mapping incoming user's tasks to available cloud resources (e.g. Virtual machines) within the suitable period of time for best resource utilization and good overall performance of cloud system [8] based on the nature and characteristics of the given scheduling algorithm. The overall performance of the scheduler depends on many parameters; reduction of makespan, best resource utilization, good distribution of workload among resources, waiting time, concurrent execution between small and large length tasks, etc. Another role in the task scheduling that important is the Meta-Task. It is a collection of tasks from different users that received by the system, here is the cloud provider. Meta tasks may be share some characteristics from each other or have similar kind of attributes [9].

The scheduling process can be generalized into three categories; Resource discovering and filtering; in this stage, the broker in datacenter contacts and provides the meaningful information about resource's status such as current availability, gathers information statistics and status of current cloud resources. Resource selection; this is called deciding phase in which the cloud resource is determined based on the given information and parameters of both tasks and resources. Task submission; in this phase, the task is transferred to the selected resource to be executed and scheduled [1].

There are various classification of task scheduling and load balancing algorithms; traditional, heuristic, Meta heuristic, etc. [10]. Task scheduling is considered as problem called NP-Complete and heuristic algorithms are best and used to find optimal solutions to solve that problem [6] [11] [12].

In addition, every user wants his task is completed with the minimum completion time, so the good scheduler is that performs the scheduling and allocation of users concurrently and fairly without starvation for some tasks or users [13]. The following section discusses several of heuristic algorithms, advantages and disadvantages.

III. HEURISTIC TASK SCHEDULING ALGORITHMS

Heuristic algorithms are types and applications of the batch mode and they are suitable algorithms in cloud task scheduling. The heuristic algorithms depend on completion time (also called makespan) of schedule. Examples of heuristic algorithms are MET, OLB, MCT, Min-Min and Max-Min task scheduling algorithms and they are discussed in the following points [12] [14].

A. Immediate Mode Scheduling

It is also named online mode. In this mode, the tasks are processed directly from the waiting queue [9].

1) Opportunistic Load Balancing (OLB)

It is an immediate and heuristic load balancing. It was suggested by Freund [15]. The main purpose of this algorithm is to make each node busy with the workload [9]. The one of its advantages is the simplicity, it executes and schedules the tasks randomly in which allocates the unexecuted task to the currently available resource [16]. It does not care about current execution time and completion time of nodes, ready time of cloud resources [15], so for this reason, tasks in this algorithm are processed slowly [16], provides us with bad result [17], and led to poor makespan [6] [18].

2) Minimum Execution Time (MET)

It is known as Limited Best Assignment (LBA) and it is an immediate and heuristic algorithm that was proposed by Armstrong [15]. The concept of this algorithm is to assign each task to its best resource with minimum execution time without considering the resource availability and its status at that time. If two resources have the same MET, then one of them is chosen randomly [4]. It is used in the SmartNet applications and several researches. It takes O(mn) for assigning the tasks, where m and n symbols are number of resources and tasks respectively. The problem of this algorithm; it does not consider the resource availability, this is lead to starvation and high load imbalance in which the incoming workload does not distributed evenly [19][20][15] [18] [4]. The Execution Time (ET) is calculated according to the following formula [21]:

$$ET_{ij}(T_i, R_j) = (Length_i / Power_j) \quad \ldots\ldots (1)$$

Where $Length_i$ is the length of task T_i measured in Million Instruction (MI), and $Power_j$ is the processing power of resource R_j measured in Million Instruction Per Second (MIPS).

3) Minimum Completion Time (MCT)

It is one of the heuristic algorithms that proposed by Michalewicz and it is a hybrid of both OLB and MET to solve their limitations [15] [18]. It is an immediate mode scheduling in which executes one task at a time and schedules every task to the resource with the earliest or least completion time of this task. Like MET, the resource is selected randomly when there are more than two resources have same MCT [4]. It takes O(mn) time for scheduling the user's tasks. The Completion Time (CT) is calculated by summation the execution time and the ready time of the resource to be available for the following task, as shown in the following equation:

$$Completion\ Time\ (CT_{ij}) = Execution\ Time\ (ET_{ij}) + Ready\ Time\ (R_j) \quad \ldots..(2)$$

Where ETij is the expected time for execution task Ti on resource Rj, and Rj is the ready time of resource j at which it finishes any previously assigned tasks.

Furthermore, there are some tasks may be executed on the resource that do not has minimum execution time for those tasks [18]. This is leads to poor utilization of resources, imbalance of workload and high makespan (e.g. increasing the maximum completion time of cloud system).

B. Batch Mode Scheduling

In this mode, the jobs are processed at a given time as batches [9].

1) Min-Min Heuristic Algorithm

It is one of batch mode heuristic algorithm that is preferred in the grid and heterogeneous computing systems and cloud computing. In this technique, the cloud manager deals with tasks based on their execution and completion times. Min-Min starts with a set Meta Task (MT) queue contains all of incoming unexecuted tasks and it works in two stages; in the first stage, the Completion Time (CT) is calculated for every unscheduled task in MT queue and Minimum Completion Time (MCT) is determined from the completion time matrix. In the second stage, it searches for task with MCT and schedules it to the corresponding resource. After that, the assigned task to that resource is removed from the MT queue; both completion time of all remaining tasks and ready time of resource are updated. This process is repeated until all of tasks in MT are scheduled to the resources [4] [17]. The main drawback of Min-Min, it starves long length tasks, especially if the small tasks are much more than larger ones, in this case it leads to imbalance of workloads, poor resource utilization, increased makespan [6][8][17][22].

2) Max-Min Heuristic Algorithm

It is same as the Min-Min, but the difference only in the second stage, Max-Min chooses a task with the maximum completion time instead of choosing task with the minimum completion time as in Min-Min, and mapped it to the corresponding cloud resource with MCT. For all remaining unexecuted tasks, the ready time is again updated and the completion time is calculated. The same procedure is repeated until all tasks are scheduled and MT queue gets empty. Both of the heuristic algorithms (Min-min and Max-Min) are applied in the small distribution systems. The heuristic complexity of them is O(mn2), where n is the number of independent tasks and m is the number of cloud resources [9][22][23].

Max-Min is coming to solve some issues in the preceding heuristic algorithms; it is better in makespan and executes tasks concurrently [12] but its main drawback, it gives high priority to large size tasks at the beginning of scheduling process [24]. Hence, when there is huge number of longer tasks with few small length tasks, this lead to starvation of small tasks in which they are waited more time to be scheduled [4][14]. The Pseudo code of Min-Min and Max-Min algorithms is shown in the following table:.

Table-I: Pseudo code for Min-Min algorithm [4]

NOTE: FOR MAX-MIN, INSTEAD OF UNDERLINED WORD (MINIMUM), REPLACE IT WITH (MAXIMUM)
// Stage 1: Calculation the Minimum CT of Each Task (Ti)
1. For all tasks (Ti) in the Meta Task set (MT)
2. For all resources (R_i) in the resources set (R)
3. Compute CTij = ET_{ii} + R_i, (for each task in all resources).
4. End For
5. End For
// Stage 2: Assigning/Scheduling Task (Ti) with Minimum Completion Time (CT) to Resource (R_i) that gives MCT.
6. Loop until all tasks in (MT) have been mapped.
7. For every task T_i in MT queue,
8. Find the task T_i with **Minimum** Completion Time (CT_{ii}) and the resources that obtain it.
9. Find the MCT and the resource $\mathbf{R_i}$ that obtains it.
10. Assign Ti to the resource R_i that gives the MCT.
11. Delete the task Ti from the MT Queue.
12. Update the resource Ready Time (RT_i).
13. Update CT_{ii} for all remaining unmapped tasks in (MT)
14. End Loop

3) Other Heuristic Algorithms

The Min-Min schedules the small tasks first on the fastest execution resources, so these resources are becoming heavily while the other resources may be idle [25]. Hence, the new algorithm called Load Balance Min-Min (LBMM) is introduced and summarized by the authors in paper [25]. It works in two phases; first the Min-Min is executed. In the second phase, the resource with the highest makespan and heavy workload is selected and reassigns some of their loads to the light resources. For that selected resource, task Ti with minimum execution time is chosen. After that, calculating the completion time (CTij) of Ti on all other resources in the schedule, the maximum CT of Ti is compared with makespan that produced from Min-Min. If it is less than the makespan, this task Ti reassigned to the resource that produces it, now the ready time of both (old and new) resources are updated. Otherwise, the next maximum CT of Ti is selected and the process is repeated again until all resources and tasks are considered for rescheduling process. It is best than the traditional Min-Min in terms; average of resource utilization and makespan when the tasks are small, but still bad for other situations [6][9][25].

R. Vijayalakshmi [6], the author proposed a new heuristic algorithm in grid computing. It performs Min-Min, then it depends on calculation the average of completion time, and rescheduling process by searching the task with maximum of average completion time and compare it with makespan that produce from Min-Min. it is best compared to Min-Min in terms of resource utilization and makespan. Prerit Chawda [18], the author provides an Improved Load balancing Min-Min Task Scheduling Algorithm (ILBMM). It executes the Min-Min first and then reschedules the least length task on heavy resource. The result shows that the proposed algorithm is better than Min-Min in terms of completion time of tasks, improving load balancing of resources. George Amalarethinam. D.I [12], the author proposed a new heuristic task scheduling in a heterogeneous Grid computing named it as Max-min Average algorithm to enhance the two important parameters in grid computing; decreasing the makespan and reducing the idle time. The proposed algorithm is done in two phases; the first Max-Min is executed. In the second phase, the mean of Completion time (meanCT) is taken for all resources. The resource whose CT is greater than the meanCT is selected, then rescheduling the tasks from that selected resource to the resource whose CT is less than the meanCT. Neha Sharma [26], the author was proposed a new hybrid algorithm that based the two algorithms Min-Min and Max-Min. It calculates the average of all of tasks in the Meta task (MT), say (AvgMT). Then, collecting tasks that are less than the AvgMT in a separated queue and applying the Max-Min. In the other hand, collecting tasks that are more than that average and applying Min-Min on them.

The result shows that it is better than the two traditional algorithms in terms of makespan, load balancing, and average of resource utilization.

J.Y Maipan-uku [27] was proposed a hybrid algorithm named Max-Average Scheduling Algorithm and depends on both the Min-Min and Max-Min traditional algorithms. First, order tasks in ascending order, then calculating the average of completion time for all tasks (AverageCT) and compare; if AverageCT is less than or equal to the smallest length task, the scheduler selects from rear of MT queue, otherwise it selects from the beginning of that queue. It is good for executing tasks quickly and efficient load balancing.

IV. THE PROPOSED ALGORITHM

Each task scheduling algorithm has its own advantages and disadvantages. Min-Min scheduling policy gives poor performance when the number of small length tasks is more than the number of longer ones, and the Max-Min is not good when the heavy tasks are much more than the light tasks. Hence, Max-Min leads to starvation of small size tasks in which it executes them at the end of scheduling process, and Min-Min delays the larger tasks.

To address this issue, the HAMM hybrid algorithm has been proposed. It provides better in most of optimization parameters; decreasing makespan, best utilization of resources, efficiently balancing of workloads among resources, enhance average of waiting time, and best concurrent execution between small and long length tasks.

To describe this algorithm, the following table describes the notations and symbols that are used in the proposed algorithm.

Table-II: Notations Used in The Proposed Algorithm

Notation	Description
MT	*Meta Task, MT = {T1, T2, T3, …, Ti}.*
TL_i	*length of task Ti measured in Million Instruction (MI).*
TLC_{min}	*the total number of tasks that are less than or equal to the AvgTL*
TLC_{max}	*the total number of tasks that are greater than AvgTL*
RU_i	*Resource Utilization of Resource Rj.*
ARU	*Average of Resource Utilization for overall system.*

The HAMM begins with calculating the average of task's length for all tasks in the MT queue (AvgTL), according to the following formula:

$$AvgTL = SUM\ (TL_1, TL_2, \ldots TL_n)\ /n, \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Where, n is the total number of all tasks in the MT at this time. After calculating the AvgTL, two empty counters are created, first counter is known as Minimum Task Length Counter (TLCmin) for counting all tasks that have length smaller than or equal to the AvgTL, and the second counter is known as Maximum Task Length Counter (TLCmax) for counting all tasks that have length greater than the AvgTL. Then, choosing task Ti from MT queue and compare its length with that average; if TLi is smaller than AvgTL, the counter TLCmin is increased with one degree, otherwise the counter TLCmax is increased with one. This process repeats until all the tasks in MT have been compared. After that, comparing between these counters, if TLCmin is greater than or equal to the TLCmax, the scheduler in the proposed algorithm performs the Max-Min task scheduling algorithm. Otherwise, the Min-Min is selected. The task Ti that scheduled is removed from the MT queue, and both of the ready time of resource and completion time of all remaining unmapped tasks are updated. This process is repeated each time until the Meta Task (MT) gets empty.

The following figure (1) and table (3) show the flowchart and Pseudo Code of proposed (HAMM) Algorithm.

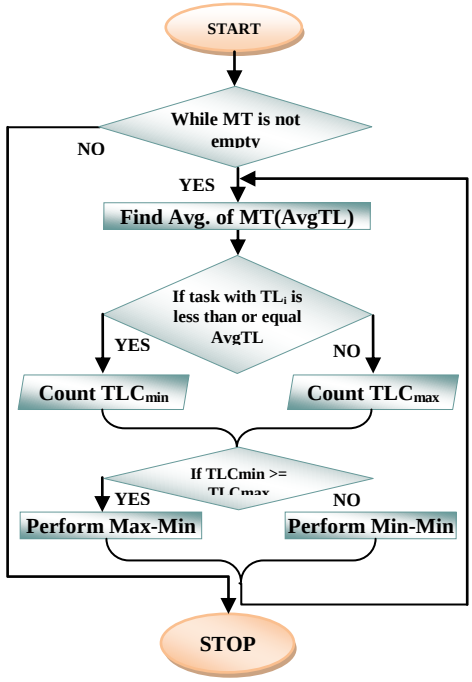

Fig. 1. Flow Chart of the Proposed Algorithm

By counting the task's number below and after the average of all tasks in MT queue; this condition works efficiently to monitor the sizes of tasks, and can predicate the number of small size tasks and long length tasks, then, choosing the suitable algorithm either Max-min or Min-Min.

Table-III: Pseudo Code for Proposed algorithm

PSEUDO CODE FOR PROPOSED (HAMM) ALGORITHM:

1. **For** all tasks T_i in the Meta Task Set (MT),
2. **Compute** Average of length of all tasks in MT, say AvgTL, according to formula 3.
3. **Count** all tasks that are less or equal to the AvgTL, into TLC_{min} counter.
4. **Count** all tasks that are greater than AvgTL into TLC_{max}.
5. **If** TLC_{min} is greater than or equal to the TLC_{max}, select *Max-Min algorithm.*
6. **Else**, Select *Min-Min* task scheduling algorithm.
7. **End if** Condition
8. **End For**

When, TLCmin is greater than the TLCmax, this means the small length tasks are greater than longer ones. In this case, the Min-Min is not suitable and it starves the long tasks, increasing their waiting time and still executes only small tasks until all of them completed.

Hence, the Max-Min is better than the Min-Min at this situation, so the HAMM chooses Max-Min to begin with the longest length task. At the next loop, the condition is tested to choose either Max-Min or Min-Min according to the length of all remaining unscheduled tasks in the Meta Task (MT) queue. In case of, TLCmin is less than the TLCmax, this means the long tasks are more than small ones, then the scheduler in the proposed algorithm selects the Min-Min at this situation to starts with smallest length task, because the Max-Min always executes from longest tasks leads to starve the small tasks in which they will wait more time until all of long length tasks have been executed. At the next time the condition is tested to choose either Max-Min or Min-Min.

V. SIMULATION AND PARAMETERS USED

For evaluating the proposed algorithm in the virtual environment, the simulation tool known as CloudSim has been used. The following paragraphs discuss the CloudSim definition, its entities, the simulation scenarios, and the performance parameters that are used in this paper.

A. CloudSim Simulation Tool

It is a toolkit simulation tool for modeling and simulation of the cloud computing environment; virtualized cloud-based data center including flexible configuration of hardware, software, memory, bandwidth and storage [23]. It provides the following advantages:

- Easy to use and basic tool in cloud computing for modeling and simulating the huge scale infrastructure or data center.
- Adaptability feature for switching between space-shared and time-shared allocation policies of processing cores to virtual services.
- Strongly supporting the virtualization engine, such as creating and managing multiple, co-hosted and independent virtual services on physical data center resources.
- Self-contained platform for simulating and modeling data centers, service brokers, allocation and scheduling policies.

There are some environments in cloud computing; homogeneous environment, in this environment, a fixed specification of the resources or virtual machines are given for checking the performance of task scheduling algorithms in cloud computing, while changing in the number of cloudlets. Heterogeneous environment; in this type, the specification of the resources or virtual machines are selected randomly with different MIPS, RAM and Bandwidth, for checking the performance of task scheduling algorithms in cloud computing [20].

B. Entities in CloudSim Architecture

There are several entities or classes in the CloudSim for implementation the task scheduling policies, some of them are mentioned in the following points [9][23]:

- **Cloudlets**

It is the user's requests incoming to class provider and executed in the power data center by the data center scheduler/broker according to the scheduling policy [29]. It is also called the tasks or jobs in the cloud environment.

- **Cloud Information Service (CIS)**

It contains the meaningful information about cloud resources, its status and resource's availability. CIS helps the cloud broker in monitoring and scheduling process.

- **Cloud Broker**

Cloud broker plays very important role between the user requests and cloud resources. It takes the required information from CIS unit. The scheduling policies occur here at this entity, so dividing the jobs into tasks and map them to cloud resources according the optimization parameters such as availability of resource and user's need. After scheduling process, the broker takes the result and backs it to user.

- **Data center (DC)**

It is the infrastructure of cloud provider in which considered as the house of computing devices, hardware, software, storage and network. Datacenter characteristics class uses for providing the information regarding the cloud resources in the data center [23].

- **Host**

This class is used for modeling physical resources such as sharing the memory, bandwidth, and processing power among virtual machines according to given policy.

- **Virtual Machine (VM)**

The VM class is responsible for allocating the processor elements among the virtual machines. In other words, it is used for modeling virtual machine in which specifies all of the virtual machines on hosts. The figure (2) shows these entities and relationship between them as detrained in the cloudSim tool.

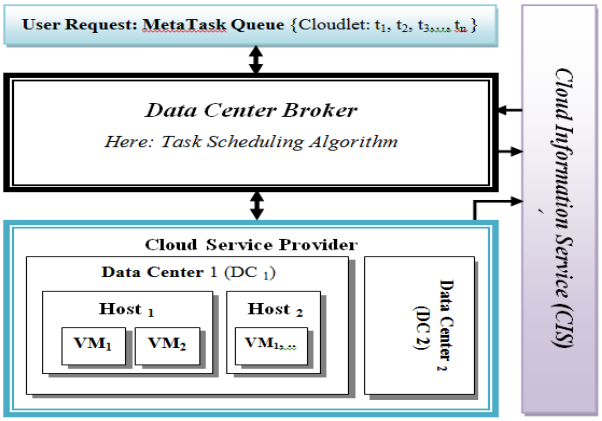

Fig. 2: CloudSim Entities and their relationship[9]

C. CloudSim Scenarios and Configuration

To evaluate the performance parameters of the proposed algorithm compared to the Min-Min and Max-Min algorithms, there are three different scenarios have been taken in the simulation experiment and analysis as shown in the following points.

- Scenario I: A few short tasks along with many long tasks; i.e. the case where Min-Min is better than Max-Min in some of the optimization parameters.
- Scenario II: Many short length tasks along with few long ones; i.e. the case where Max-Min outperforms Min-Min.

- Scenario III: Length of tasks are randomly determined.

The number of virtual machines remains constant (VM=10) for all of the simulation scenarios, and the number of user's cloudlets is chosen to be 1000. Then, the length of these cloudlets is variable and changing according to the previous three different scenarios. The following tables (table 4 and table 5) show the configuration in the CloudSim simulation tool for cloud resources.

Table-IV: Virtual Machine's Configuration In Each Host

Sr. No.	Characteristic	Value
1	Number of VMs	(10)
2	Size/Processing speed (in MIPS)	(1000 – 10000)
3	No. of CPUs/cores	1
4	RAM	2048 (in MB)
5	Bandwidth	1000 (in mbps)

The following table represents the host configuration in each data center.

Table-V: Host Configuration

Sr. No.	Characteristic	Value
1	RAM	2048 * 2 (in MB)
2	Bandwidth	1000 (in mbps)
3	Storage	1000000 (in MB)

D. Performance Parameters

The performance of result analysis is evaluated according to the following Parameters:

- **Makespan (MS)**

It is the maximum of total completion time for all tasks in the cloud system. It is also measured by the maximum of completion time on the resource. The scheduling system is better when the makespan is low [26][30]. It is shown in the following formula [6]:

$$MS = Max\ (CT_{ij}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Where CT_{ij} *is the completion time of tasks* T_i *on resource* R_j.

- **Average Of Resource Utilization**

Another optimization criterion in task scheduling is the resource utilization which is used to measure algorithm efficiency [27]. It is the measure for reducing the idle time of resource R_j or it is the time percentage that the machine R_j is busy during the allocation and scheduling process [31]. The average resource utilization of the overall system is taken as the average of resource utilization for all resources R={1, 2, 3, .., M} in the system. For good scheduler system, the high average resource utilization must be achieved [26].

- **Load Balancing (LB)**

It is the best distribution of workloads between all cloud resources in efficient manner. It should be high for efficient and best task scheduling algorithm [26] [32].

- **Average OF Waiting Time (AWT)**

It is another optimization parameter of task scheduling algorithms. It is the time amount that waited by the next task to get cloud resource. It is calculated as the sum of times that the task spent in the Meta-task queue. Then, the waiting time average is the summation of all of the waiting times for all tasks in the schedule [33].

- **Concurrent Execution of Tasks**

The opportunity of concurrent execution of tasks between small and large tasks is considered as the important parameter in which preventing the starvation state for certain tasks [34].

VI. RESULT AND DISCUSSION

To verify the proposed algorithm, the simulation was done using Cloudsim Simulator to compare proposed algorithm with other two algorithms; Min-Min, Max-Min based on the following parameters; Makespan, average of resource utilization, load balancing, average of waiting time and concurrent execution between heavy and light tasks. The following sub sections show the performance analysis of the algorithms; Min-Min, Max-Min and proposed (HAMM) algorithm for the given three scenarios. The result is represented in two different forms; tabular and graphically. The simulation has been calculated for each scenario for 10 times and after that the average has been taken.

A. Scenario (I)

In this scenario, the different parameters have been tested:

1) Makespan Parameter

From table (6) and figure (3), it can be seen that the simulation result of the proposed HAMM algorithm is better than both Min-min and Max-Min algorithms in term of makespan parameter.

Table-VI: Makespan Optimization Parameter

Min-Min	Max-Min	HAMM
45.87	45.27	44.74

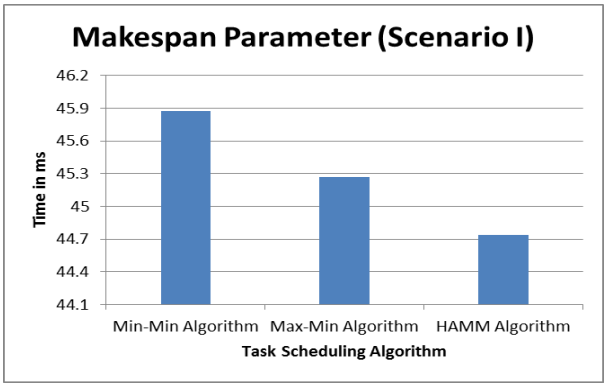

Fig. 3. Makespan Parameter in Scenario I

2) Average of Waiting Time parameter

The average of waiting time parameter is depicted in the table (7) and figure (4). The simulation result shows that the proposed algorithm is best in this parameter compared to the other algorithms; Min-Min and Max-Min.

Table-VII: Average of Waiting Time Optimization Parameter

Min-Min	Max-Min	HAMM
45.2	45.27	44.69

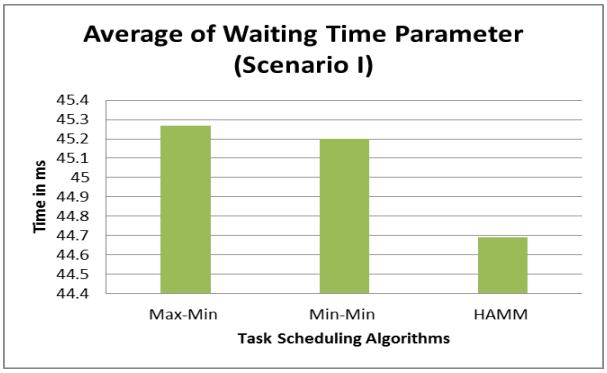

Fig. 4: Average of Waiting Time Parameter for Scenario I

3) Average of Resource Utilization (ARU) and Load Balancing Parameters

The average of resource utilization (ARU) and load balancing parameters are showed in the table (8) and fig. (5). The Min-Min is less than the other two algorithms in these parameters. The others; Max-Min and proposed (HAMM) are best and have the same degree in the average of resource utilization and load balancing parameters.

Table-VIII: ARU and Load Balancing Optimization Parameters

PARAMETER	MIN-MIN	MAX-MIN	HAMM
ARU	0.99	1.00	1.00
LOAD BALANCING	0.99	1.00	1.00

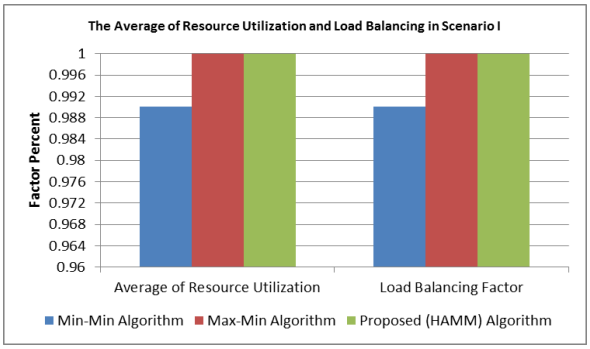

Fig. 5: Average of Resource Utilization and Load Balancing Parameters in Scenario I

B. Scenario (II)

In this scenario, the result of simulation has been discussed for different parameters as follows:

1) Makespan and average of waiting time parameters

Both of the table 9 and figure 6 show the simulation result of the makespan and average of waiting time parameters for Min-Min, Max-Min and proposed algorithm.

Table-IX: Makespan and Average of Waiting Time Parameters

PARAMETER	MIN-MIN	MAX-MIN	HAMM
MAKESPAN	8.03	7.81	7.76
AWT	7.79	7.81	7.76

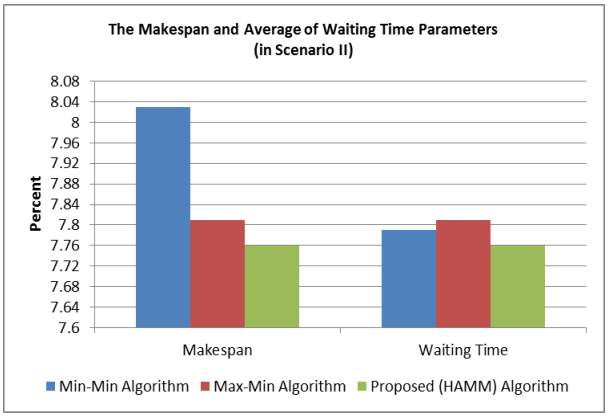

Fig. 6. Makespan and Average of Waiting Time Parameters (Scenario II)

From the above result, the proposed algorithm is best when compared to the Min-Min and Max-Min in the makespan and average of waiting time parameters.

2) Average of Resource Utilization (ARU) and Load Balancing Parameters.

The simulation result for average of resource utilization and load balancing parameters has been represented in table 10.

Table-X: ARU and Load Balancing Optimization Parameters

PARAMETER	MIN-MIN	MAX-MIN	HAMM
ARU	0.97	1.00	1.00
LOAD BALANCING	0.97	1.00	1.00

The result shows the proposed algorithm is better than the Min-Min, and it is best and same as the Max-Min in the utilization of cloud resources and load balancing parameters. The simulation result also represented graphically in the figure (7).

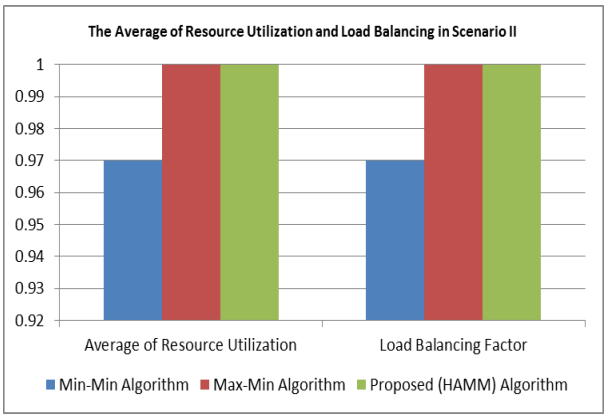

Fig. 7. Average of Resource Utilization and Load Balancing Parameters in Scenario II

C. Scenario (III)

The simulation result has been done in this scenario for the following optimization parameters:

1) Makespan and average of waiting time

Both of the following table (11) and figure (8) show the simulation result for Min-Min, Max-Min and proposed HAMM algorithm. According to that result, the proposed algorithm is better than the Min-Min and Max-Min in terms of makespan and average of waiting time optimization parameters.

Table XI: Makespan and Average of Waiting Time Parameters

PARAMETER	MIN-MIN	MAX-MIN	HAMM
MAKESPAN	9.24	9.13	8.96
AWT	9.11	9.13	8.95

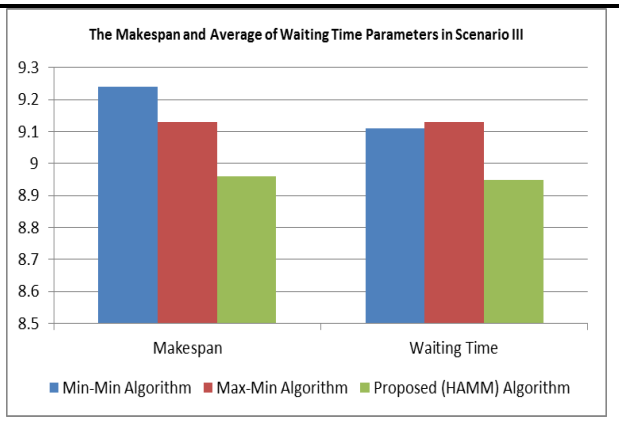

Fig. 8. Makespan & Average of Waiting Time Parameters (Scenario 3)

2) ARU and Load Balancing Parameters

The simulation results for average of resource utilization and load balancing parameters have been represented in the table (12).

Table-XII: ARU and Load Balancing Optimization Parameters

PARAMETER	MIN-MIN	MAX-MIN	HAMM
ARU	0.98	1.00	1.00
LOAD BALANCING	0.98	1.00	1.00

From the result, the HAMM is better than the Min-Min, and both Max-Min and proposed (HAMM) are best and have the same degree in the average of resource utilization and load balancing parameters. The simulation result also represented graphically in the figure (9).

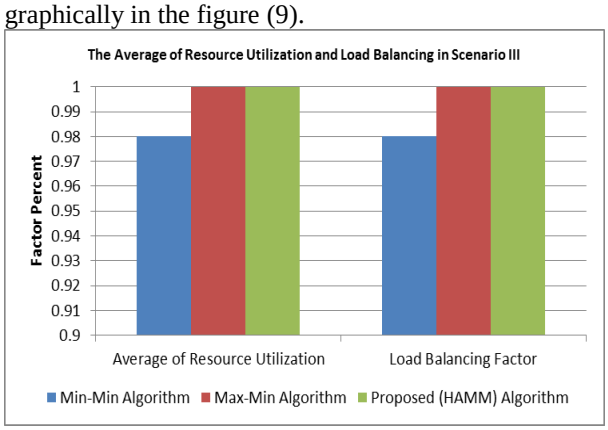

Fig. 9: Average of Resource Utilization and Load Balancing Parameters in Scenario III

D. The Concurrent Execution Between Heavy Tasks and Light Tasks (For all three scenarios)

According to the simulation result in all scenarios, the Min-Min executes the cloudlets in ascending order in which the scheduling process begins with the smallest cloudlet towards the longest task. The Max-Min schedules tasks in descending order; begins with longest task and process one by one until finally reach to the smallest task. In both algorithms, they lead to starvation. In the Min-Min, the large tasks wait until the final of scheduling process while in the Max-Min, the small tasks are waited and executed finally. The following figures (10) and (11) show the qualitative execution order of Min-Min and Max-Min respectively.

```
SN[195] The Cloudlet No.[101] with 885 is bind to VM[5] with MCT: 1.6171666666666664
SN[196] The Cloudlet No.[410] with 890 is bind to VM[9] with MCT: 1.6304999999999998
SN[197] The Cloudlet No.[789] with 892 is bind to VM[6] with MCT: 1.6418571428571431
SN[198] The Cloudlet No.[428] with 897 is bind to VM[8] with MCT: 1.6505555555555558
SN[199] The Cloudlet No.[677] with 902 is bind to VM[4] with MCT: 1.6585999999999999
SN[200] The Cloudlet No.[232] with 909 is bind to VM[2] with MCT: 1.6609999999999998
SN[201] The Cloudlet No.[202] with 913 is bind to VM[7] with MCT: 1.6841250000000003
SN[202] The Cloudlet No.[960] with 920 is bind to VM[3] with MCT: 1.6887499999999998
SN[203] The Cloudlet No.[123] with 924 is bind to VM[9] with MCT: 1.7228999999999999
SN[204] The Cloudlet No.[212] with 925 is bind to VM[8] with MCT: 1.7533333333333336
SN[205] The Cloudlet No.[326] with 926 is bind to VM[5] with MCT: 1.7714999999999996
SN[206] The Cloudlet No.[732] with 928 is bind to VM[6] with MCT: 1.7744285714285717
SN[207] The Cloudlet No.[658] with 945 is bind to VM[7] with MCT: 1.8022500000000004
SN[208] The Cloudlet No.[554] with 954 is bind to VM[9] with MCT: 1.8182999999999998
```

Fig. 10. Execution Order of Tasks in the Min-Min algorithm

```
SN[152] The Cloudlet No.[819] with 4199 is bind to VM[7] with MCT: 13.13925
SN[153] The Cloudlet No.[861] with 4196 is bind to VM[8] with MCT: 13.201888888888893
SN[154] The Cloudlet No.[889] with 4191 is bind to VM[9] with MCT: 13.2506
SN[155] The Cloudlet No.[624] with 4186 is bind to VM[0] with MCT: 13.382
SN[156] The Cloudlet No.[983] with 4185 is bind to VM[1] with MCT: 13.560500000000001
SN[157] The Cloudlet No.[155] with 4183 is bind to VM[2] with MCT: 13.602333333333334
SN[158] The Cloudlet No.[224] with 4175 is bind to VM[4] with MCT: 13.637999999999998
SN[159] The Cloudlet No.[349] with 4173 is bind to VM[3] with MCT: 13.6375
SN[160] The Cloudlet No.[420] with 4164 is bind to VM[5] with MCT: 13.643666666666665
SN[161] The Cloudlet No.[870] with 4162 is bind to VM[6] with MCT: 13.654571428571428
SN[162] The Cloudlet No.[447] with 4162 is bind to VM[7] with MCT: 13.659500000000001
SN[163] The Cloudlet No.[70] with 4150 is bind to VM[8] with MCT: 13.663000000000004
SN[164] The Cloudlet No.[243] with 4130 is bind to VM[9] with MCT: 13.6636
SN[165] The Cloudlet No.[680] with 4123 is bind to VM[9] with MCT: 14.0759
SN[166] The Cloudlet No.[994] with 4120 is bind to VM[8] with MCT: 14.120777777777782
SN[167] The Cloudlet No.[993] with 4118 is bind to VM[7] with MCT: 14.17425
SN[168] The Cloudlet No.[482] with 4116 is bind to VM[6] with MCT: 14.242571428571427
SN[169] The Cloudlet No.[231] with 4115 is bind to VM[5] with MCT: 14.329499999999998
SN[170] The Cloudlet No.[278] with 4108 is bind to VM[4] with MCT: 14.459599999999998
```

Fig. 11: Execution Order of Tasks in the Max-Min algorithm

The proposed (HAMM) algorithm outperforms the Min-Min and Max-Min, it is concurrent between large length tasks and small length tasks efficiently and according to the context situation of the existing tasks and their sizes in Meta Task (MT) queue, this is overcome the starvation state that found in both Min-Min and Max-Min algorithms. The figure (12) which is screenshot printed from the CloudSim result shows the best concurrent execution, and this is good and efficient in all three scenarios for the proposed (HAMM) algorithm when compared to others; Min-Min and Max-Min.

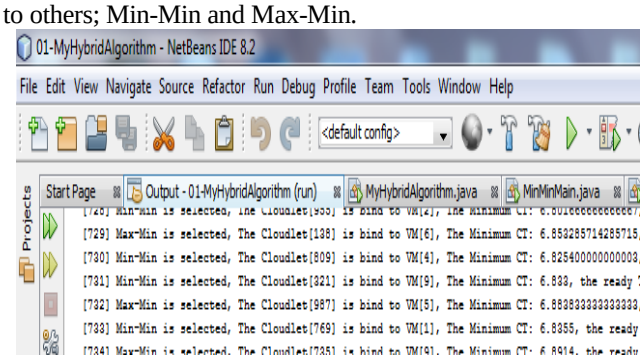

Fig. 12: Execution Order of Tasks in the proposed (HAMM)

VII. CONCLUSION AND FUTURE WORK

Task scheduling of incoming user's tasks among the cloud resources is the main concern in the cloud computing to achieve better performance of cloud system. In this paper, a new hybrid algorithm has been proposed that depends on both Min-Min and Max-Min traditional algorithm. The Cloudsim simulation tool has been used to evaluate this proposed algorithm. The result of proposed algorithm is better than the other algorithms; Min-Min and Max-Min.

It provides best resource utilization, efficient load balancing, better average waiting time, good makespan, and best concurrent execution between long length tasks and small length tasks. In future work, the proposed algorithm can be improved by using heuristic algorithms characteristics in dynamic task scheduling algorithms and combined these algorithms with artificial intelligence technologies.

REFERENCES

1. Shah Mihir N1, Asst. Prof. Yask Patel, " A Survey Of Task Scheduling Algorithm In Cloud Computing", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Vol. 4, Issue 1, Jan. 2015.
2. Yatendra Sahu and R.K. Pateriya, Cloud Computing Overview with Load Balancing Techniques, International Journal of Computer Applications, Volume 65– No.24, March 2013.
3. J. Kok Konjaang, J.Y. Maipan-uku, and Kumangkem Kennedy Kubuga, " An Efficient Max-Min Resource Allocator and Task Scheduling Algorithm in Cloud Computing Environment", International Journal of Computer Applications, vol. 142, no. 8, pp. 0975 – 8887, May 2016.
4. J.Y Maipan-uku, I. Rabiu, and Dr. Amit Mishra, "IMMEDIATE/BATCH MODE SCHEDULING ALGORITHMS FOR GRID COMPUTING: A REVIEW", International Journal of Research – GRANTHAALAYAH, Vol.5, no. 7, ISSN- 2350-0530(O), ISSN-2394-3629(P), Jul. 2017.
5. Riyadh I. Louis, A hmed I. Saleh, Mohammed F. AL Rahmawy, et al., "A Best Effort Heuristic Algorithm for Scheduling Timely Constrained Tasks i n the Cloud", International Journal of Scientific & Engineering Research, Volume 7 , Issue 2, February-2016.
6. R. Vijayalakshmi and V. Vasudevan, "Static Batch Mode Heuristic Algorithm for Mapping Independent Tasks in Computational Grid", Journal of Computer Science, 2015.
7. Mahfooz Alam and Zaki Ahmad Khan, "Issues and Challenges of Load Balancing Algorithm in Cloud Computing Environment", Indian Journal of Science and Technology, Vol 10, July 2017.
8. Jamilu Yahaya Maipan-uku, A. MISHRA, A. ABDULGANIYU, et.al, "An Extended Min-Min Scheduling Algorithm in Cloud Computing", Vol. 5. no. 2, 2018.
9. Puneet Banga and Sanjeev Rana, PhD, "Heuristic based Independent Task Scheduling Techniques in Cloud Computing: A Review", International Journal of Computer Applications (0975 – 8887) Volume 166 – No.1, May 2017.
10. Rasha A. Al-Arasi1 and Anwar Saif, " Task scheduling in cloud computing based on metaheuristic techniques: A review paper", EAI Endorsed Transactions on Cloud Systems, DOI: 10.4108/eai.13-7-2018.162829, 2020.
11. George Amalarethinam. D.I, and Vaaheedha Kfatheen .S, "Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems", International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012.
12. Belal Ali Al-Maytami, Pingzhi Fan, Abir Hussain, et al., "A Task Scheduling Algorithm with Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing", IEEE Access, Vol. 7, 2019.
13. Wang Yizhen, Sun Yongqiang and Sun Yi, Task Scheduling Algorithm in Clo ud Computing Based on Fairness Load Balance and Minimum Completion Time, 4th National Conference on Electrical, Electronics and Computer Engineering (NCEECE 2015), Published by Atlantis Press, 2016.
14. Ahmed Subhi Abdalkafor and Khattab M. Ali Alheeti, "A hybrid approach for scheduling applications in cloud computing environment", International Journal of Electrical and Computer Engineering (IJECE), Vol. 10, No. 2, pp. 1387~1397, April 2020.
15. Somayeh Taherian Dehkordi and Vahid Khatibi Bardsiri, "TASA: A New Task Scheduling Algorithm in Cloud Computing", Journal of Advances in Computer Engineering and Technology, 1(4) 2015.
16. Aanjey Mani Tripathi*, Sarvpal Singh, A literature review on algorithms for the load balancing in cloud computing environments and their future trends, COMPUTER MODELLING & NEW TECHNOLOGIES 2017 21(1) 64-73, 2017.
17. Dharmesh Kashyap, and Jaydeep Viradiya, A Survey Of Various Load Balancing Algorithms In Cloud Computing, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, Vol. 3, Issue 11, November 2014.
18. Prerit Chawda and Partha Sarathi Chakraborty, "An Improved Min-Min Task Scheduling Algorithm for Load Balancing in Cloud Computing", International Journal on Recent and Innovation Trends in Computing and Communication, ISSN: 2321-8169 Volume: 4 Issue: 4, 2016.
19. I. Kurochkin and E.A. Gerk, "MODELING OF TASK SCHEDULING IN DESKTOP GRID SYSTEMS AT THE INITIAL STAGE OF DEVELOPMENT", Proceedings of the VIII International Conference "Distributed Computing and Grid-technologies in Science and Education" (GRID 2018), Dubna, Moscow region, Russia, September 10 - 14, 2018.
20. SyedHamidHussain Madni, MuhammadShafie Abd Latiff and MohammedAbdullahi, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment", PLOS ONE| https://doi.org/10.137 1/journal.pone.0176321 May 3, 2017.
21. P. Keerthika and P. Suresh, "A Multiconstrained Grid Scheduling Algorithm with Load Balancing and Fault Tolerance", Hindawi Publishing Corporation, Scientific World Journal, Volume 2015.
22. Settu Bharti and Naseeb Singh, "LOAD BALANCING ISSUES AND ITS SOLUTION IN CLOUD COMPUTING: A REVIEW", INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY, Vol.14, No.3, April, 2015.
23. R.Jemina Priyadarsini rockiam, Performance Evaluation of Min-Min and Max-Min Algorithms for Job Scheduling in Federated Cloud, International Journal of Computer Applications, Volume 99–No.18, August 2014.
24. Manjot Kaur Bhatia, "Task Scheduling in Grid Computing: A Review", Advances in Computational Sciences and Technology, ISSN 0973-6107 Volume 10, Number 6, pp. 1707-1714, 2017.
25. Riddhi Varude, Ishita Shah, Mukesh Bhandari, "Enhanced Load Balanced Min-Min Algorithm in Cloud Computing", International Journal of Computer Sciences and Engineering, Open Access, Volume 4, Issue 4, 2016.
26. Neha Sharma, Dr. Sanjay Tyagi, Swati Atri, "HYMM: A New Heuristic in Cloud Computing", International Research Journal of Engineering and Technology, Vol. 04, no. 05, May, 2017.
27. J.Y Maipan-uku, A. Muhammed, A. Abdullah, M. Hussin, Max-Average: An Extended Max-Min Scheduling Algorithm for Grid Computing Environment, ISSN: 2180 – 1843 e-ISSN: 2289-8131 Vol. 8 No. 6.
28. Harish C. Sharma1, Meenakshi Bisht, "Best Fit Resource Allocation in Cloud Computing", Vol. 7, Issue 3, March 2019
29. Shweta Patel and Prof. Mayank Bhatt, "Implementation of Load balancing in Cloud computing thorough Round Robin & Priority using cloudSim", International Journal for Rapid Research in Engineering Technology & Applied Science Vol 3 Issue 11 November 2017.
30. Jyoti bansal, Vishu Narula, Dr. Shaveta, et al, "An Efficient Batch-Mode Scheduling Heuristic Based on Load Balancing", International Journal of Engineering and Technology (IJET), Vol. 7 No 6 Dec 2015-Jan 2016.
31. Naglaa M. Reda, A. Tawfik, Mohamed A. Marzok, et al, "Sort-Mid tasks s cheduling a lgorithm in grid computing", Journal of Advanced Research, 6 , 987–993, 2014.
32. Huankai Chen, Professor Frank Wang and Dr Na Helian, et al, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing", Conference Paper, February, 2013.
33. Navjot Kaur and Nivit Gill, "Performance Analysis Of Scheduling Algorithms In Grid Computing", International Journal of Engineering Research & Technology, Vol. 2, Issue 7, July – 2013.
34. Upendra Bhoi and Purvi N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International journal of applications or innovations in engineering and management, Volume 2, Issue 4, April 2013.

AUTHORS PROFILE

Ibrahim Ali Thiyeb was born in Taiz, Yemen in 1984. He received the B.S. degree in information technology and engineering from Aden University, Yemen, in 2009. He is currently pursuing the M.S. degree in Information Technology and Management (MIT) at Sana'a University, College FCIT, Sana'a, Yemen. Currently, He works as a tutor in the field of computer and information technology at Sana'a University. His research interest includes the development of applications, system analysis, computer programming, computer networks and cloud computing.

Dr. Sharaf Abdulhak Alhomdy, was born in Alsena , Taiz, Yemen in 20/01/1971. He has got his Ph.D. in Computer Science, Pune University, India, 2009. He is an Assistant Prof. in IT Department, Faculty of Computer and Information Technology (FCIT), Sana'a University. He was a Vice Dean for students' affairs, FCIT, Sana'a University, Yemen, 2012-2016. He is an author's of a number of papers. He was promoted to Associate Professor in June 2015, FCIT, Sana'a University, Yemen