International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Development of Load Balancing Methodology in Cloud Computing Platforms

[1]Mohammed Khairullah Mohsin, [2]Mustafa A. Fiath

Submitted: 16/11/2022 **Accepted**: 19/02/2023

Abstract: Load balancing is the process of distributing customer tasks among multiple computing resources, such as virtual machines (VMs), servers and networks. It is a major concern in cloud computing as the number of customer demanding the service is growing exponentially. An efficient load balancing approach can detect the load of the VMs proactively and assigns the customer tasks to the VMs accordingly. In this paper, we present a mechanism on load balancing in cloud using probability theory. The main aim of the proposed approach is to reduce the standard deviation of the load between the virtual machines so that they are close to zero.

Keywords: *cloud computing, load balancing, task scheduling, probability theory, resources allocation.*

1. Introduction

Cloud computing is a model for sharing resources (such as networks, servers, storage, applications, and services), software, and information to different user devices on demand [1]. Cloud computing shows continuous growth in the business community because it provides cost-cutting solutions to customers [2,10]. This explosive growth makes cloud service providers focus on efficient management of resources. There is also a need to fairly distribute customer orders to virtual machines, so that they can accommodate a larger number of customer orders without increasing the physical infrastructure [3].

Balanced task scheduling is very important in cloud computing and has been given a lot of attention in recent years. Irrespective of load balancing, service providers must define QOS (Quality Of Service) parameters such as Throughput, Availability, Flexibility, etc., to attract their customers [4]. A customer task may fall into one of two categories, either a high QoS task or a low QoS task. A high QoS task is a type of task that can be assigned to a subset of the available virtual machines, while a low QoS task can be assigned to all available virtual machines. Therefore, a task that requires a high QoS has a higher priority than a task that requires a low QoS. The current research problem in cloud computing is to schedule a mixture of tasks that require high and low QoS to a group of virtual machines with different processing capabilities, so that the load on the virtual machines is distributed fairly.

Mohamedkhiry270@gmail.com
Ministry of Education, Directorate of Education, Anbar Governorate, Iraq
azeezmustafa89@uoanbar.edu.iq
College of Medicine, Anbar University, Iraq

Many researchers have introduced load balancing algorithms to distribute tasks on computing resources. Most of these algorithms did not take into account the parameters of the quality of service provided.

In this paper, we have studied the problem of task scheduling in cloud computing based on probability theory. The proposed approach consists of two phases, in the first phase the tasks are sorted based on the availability of virtual machines. Initial loads are calculated for each virtual machine. In the second stage, tasks are allocated to the virtual machines according to the initial loads that were calculated in the first stage, and then the loads are updated for each virtual machine based on the new conditions.

2. Research justifications

Load balancing refers to the fair distribution of tasks across computing resources in cloud systems. Load balancing is a critical issue for optimizing resource utilization, improving system response time, maintaining available services, and thus improving system performance.

The importance of the research is derived from the importance of cloud computing, which constitutes the sector of the future, as well as the amount of digital data that is increasing very dramatically in the coming years. Optimal investment of computer equipment and proper load balancing leads to optimal use of computing resources, thus increasing utilization, which in turn leads to optimal investment of available resources away from the need to add new equipment at an exorbitant cost.

3. Objectives

The research aims to improve the quality and efficiency of the cloud system by developing an algorithm that achieves a balanced and stable load in the cloud. Therefore, the current load balancing algorithms were studied and analyzed and a load balancing algorithm was proposed that takes into account the size of the task and the resources allocated to each virtual machine, with the aim of improving system performance and achieving load balancing and quality the service.

4. Literature Reviews

In [4], an algorithm for load balancing was developed using the Particle Swarm Optimization (PSO) algorithm, which is a technique inspired by the collective behavior of flocks of birds. n represents the number of tasks to be scheduled, and each element within this vector is a random number between 1 and m, where m is the number of virtual machines.

The total execution time is calculated for each individual, then the individual with the lowest execution time is selected, and the tasks are assigned to the specified virtual machines within the chosen vector. The main problem with this method is the early convergence in reaching the solution if the search space is small and this solution represents the best solution within the chosen search space only and not the best solution that can be reached within the cloud data center, and if a large initial community is taken, this It will lead to a significant increase in the time to find the best individual and thus a significant increase in the waiting time for tasks until they are scheduled and thus an increase in the response time.

In [9], the Round Robin (RR) algorithm was developed in order to reduce the response time. The proposed method is based on dynamically allocating the execution time of tasks for each cycle. The time share for the first cycle is equal to the average expected execution times for the tasks. In the second cycle, the completed tasks are removed from the list and the remaining execution times for the unfinished tasks are averaged. This process is repeated until all tasks in the task list have been completed. This study relied on optimizing the scheduling of tasks that were allocated to virtual machines and relied on the process of assigning tasks to virtual machines on the static Round Robin algorithm. Thus, this algorithm improves processing time for small-sized tasks, but it does not take into account the resources allocated to each virtual machine when allocating tasks, so it may lead to Trashing or a significant increase in processing time for large-sized tasks.

In [6] a Load Balancing Decision Algorithm (LBDA) algorithm was proposed for load balancing between virtual machines within the cloud data center in order to reduce the overall execution time and response time. The mechanism of action of the proposed algorithm depends on three stages: first, the processing capacity of the virtual machines and the current load on each of them are calculated and classified into (Underload, Balance, High Balance, Overload), then the estimated execution time for the task is calculated on each virtual machine in the Underload state and the assignment of the task To the virtual machine that achieves the lowest execution time, and in the absence of virtual machines in the Underload state, the estimated execution time for the task is calculated on the virtual machines that are in the Balance state, and so on. If all virtual machines are in the Overload state, the job is queued until one of the virtual machines changes state. This algorithm depends on the processing capacity of the virtual machines during its classification process, and therefore it does not take into account other parameters such as internal storage and external storage, and this may lead to the occurrence of Trashing for some tasks if the memory is limited and insufficient to perform the task.

5. Proposed Approach

The resources allocated to each virtual machine are different and usually change dynamically according to the resources reserved by the tasks assigned to the virtual machine and the resources released when the virtual machine finishes executing a specific task. The capacity of each virtual machine is calculated in terms of the resources available to each of them as follows [5]:

$$\tau_{CPU} = \frac{P_{CPU}}{P_{Max}} \times 100\% \qquad CPU$$

$$\tau_{mi} = \frac{m_i}{m_{iMax}} \times 100\% \qquad \text{Internal Storage}$$

$$\tau_{me} = \frac{m_e}{m_{eMax}} \times 100\% \qquad \text{External Storage}$$

$$\tau_j = (\phi_1 \times \tau_{CPU}) + (\phi_2 \times \tau_{mi}) + (\phi_3 \times \tau_{me}) \qquad : \sum \phi = 1$$

Where:

τ_j : Virtual machine capability.

P_{CPU} : The processing power available within the virtual machine.

m_i : Internal storage available within the virtual machine.

m_e : External storage available within the virtual machine.

P_{Max} : The total processing power allocated to the virtual machine.

m_{iMax} : Total internal storage allocated to the virtual machine.

m_{eMax} : Total external storage allocated to the virtual machine.

ϕ : Weight parameter to adjust the degree of resource impact.

The capability of each virtual machine is constantly changing, depending on several factors:

- The capability of the virtual machine τ_j is reduced when a new task is assigned to it and the amount of deficiency equals $(1-\mu)$ depends on the percentage of resources consumed relative to the total resources allocated to this node [5]:

$$\tau_j(t+1) = (1-\mu) \times \tau_j(t) \qquad (1)$$

Where: μ Parameter to determining the ratio of consumed resources to total resources.

- The capability of the virtual machine τ_j is increased upon completion of a specific task, and the amount of increase depends on the percentage of freed resources that have been allocated to that task [5].

$$\tau_j(t+1) = (1+v) \times \tau_j(t) \qquad (2)$$

Where: υ Parameter to determine the ratio of freed resources to total resources.

The ratio of the load L_{ij} caused by task i on the virtual machine j relative to the rest of the virtual machines is calculated according to the following equation:

$$L_{i=\overline{1-n},,j=\overline{1-m}} = \frac{ET_{ij} \times (1-\tau_j)}{\sum_{k=1}^{m} ET_{ik} \times (1-\tau_k)} \qquad (3)$$

Where:

m: The number of virtual machines.

n: the number of tasks.

ET (Execution Time): The expected execution time of task i on the virtual machine j.

The Execution Time (ET) [8] on each virtual machine is calculated according to the following formula:

$$ET = \frac{TL}{Capacity * cores(T)} \qquad (4)$$

Where:

TL (Task Length): The length of the task T.

Capacity: The rate of processing capacity per core (MIPS).

Cores(T): The number of cores needed by task T.

Thus, the result of applying equation (3) to n tasks and m virtual machines is a load matrix that looks like this:

$$L_{i=\overline{1-n},j=\overline{1-m}} = \begin{bmatrix} L_{11} & \dots & L_{1m} \\ L_{21} & \dots & L_{2m} \\ L_{n1} & \dots & L_{nm} \end{bmatrix} \qquad Where \quad n >> m$$

The total load (TL) caused by all tasks is calculated on the VM_j virtual machine:

$$TL(VM_j) = \sum_{i=1}^{n} L_{ij} \qquad (5)$$

When the job is allocated to the VM_j, the loads are updated for each virtual machine. The new total load $TL(VM_j)$ of the VM_j to which the job is allocated is calculated according to the following relationship:

$$TL_{new}(VM_j) = TL_{old}(VM_j) + (1 - L_{ij}) \qquad (6)$$

So the new load for the rest of the virtual machines is:

$$TL_{new}(VM_j) = TL_{old}(VM_j) - L_{ij} \qquad (7)$$

The above two equations show that when task i is assigned to a virtual machine, the percentage load on the selected virtual machine increases, and the amount of the increase is the sum of the percentages of load generated by task i on the rest of the virtual machines. It is generated by the first task on each VM separately and the mechanism for applying the above two equations will be discussed in the next section.

The main objective of the proposed approach is to reduce the standard deviation σ of the load between virtual machines so that its value is very close to zero. Where the standard deviation σ is used to measure the extent of data dispersion from the average value [13].

$$\sigma = \sqrt{\frac{1}{m} \times \sum_{j=1}^{m} (TL(VM_j) - Average(TL))^2} \qquad (8)$$

6. Results and Discussion

In this section, we will apply two work scenarios. The first scenario will generate a load matrix randomly so that the number of tasks is a multiple of the number of virtual machines. The size of the matrix is (21X3) and includes 3 columns representing the number of virtual machines, m = 3, and 21 lines representing the number of tasks n = 21, and in the second scenario, two new tasks will be added to the previous matrix so that its size becomes (23 x 3), where the number of tasks is not a multiple of the number of virtual machines.

1) In the first scenario, Table (1) shows the matrix of randomly generated loads. Each box represents the L_{ij}

load that the Ti task causes on the VM_j in relation to the rest of the VMs, so the sum of each line is equal to one, 2) and an X indicates that the VM is not available for the task.

	VM1	VM2	VM3
T1	0.5	0.25	0.25
T2	0.45	X	0.55
T3	0.25	0.3	0.45
T4	0.6	0.2	0.2
T5	0.2	0.8	X
T6	0.75	X	0.25
T7	0.2	0.2	0.6
T8	0.15	0.7	0.15
T9	0.1	0.1	0.8
T10	X	0.6	0.4
T11	X	0.2	0.8
T12	0.65	0.35	X
T13	0.35	X	0.65
T14	0.75	0.1	0.15
T15	X	0.7	0.3
T16	0.8	0.1	0.1
T17	X	0.75	0.25
T18	0.65	X	0.35
T19	0.3	X	0.7
T20	0.3	0.3	0.4
T21	0.45	0.25	0.3
Total Load (TL)	**7.45**	**5.9**	**7.65**

Table 1*:* Matrix of Randomly Generated Loads

The Round Robin algorithm will be applied to the matrix shown in Table (1), where the algorithm starts by allocating the first task to the first available virtual machine and then moves to the next available virtual machine, and this process is repeated circularly until all tasks are allocated. Every time a specific task is assigned to one of the virtual machines, the total load of the virtual machine to which the task has been allocated is updated according to equation (6), and the total load of the rest of the virtual machines is updated according to equation (7).

- Equation (5) is applied to the data of Table (1) to calculate the total load $TL(VM_j)$ on each virtual machine:

$$TL\ (VM1)=\sum_{i=1}^{21} Li1 =7.45$$

$$TL\ (VM2)=\sum_{i=1}^{21} Li2 =5.9$$

$$TL\ (VM3)=\sum_{i=1}^{21} Li3 =7.65$$

- Task T1 in table (1) is assigned to VM1, and thus the total load of VM1 is updated according to relationship (6) as follows:

$$TL\ (VM1) =7.45+(1-0.5)=7.95$$

- The total load of the remaining available virtual machines VM2, VM3 is updated according to equation (7) as follows:

$$TL\ (VM2) =5.9-0.25=5.65$$

$$TL\ (VM3) =7.65-0.25=7.4$$

- Cycle repeating the previous steps until all tasks have been allocated.

Table (2) shows the results of applying the previous steps to the load matrix shown in Table (1) using the Round Robin (RR) algorithm, where the boxes in bold

indicate the default machine to which the task is assigned, and the values in the table show the change in the load ratio on each machine default on each assignment to a new task.

	VM1	VM2	VM3
TL	7.45	5.9	7.65
T1	7.95	5.6	7.4
T2	7.5	5.65	7.85
T3	8.25	5.35	7.4
T4	7.65	6.15	7.2
T5	8.45	5.35	7.2
T6	7.7	5.35	7.95
T7	8.5	5.15	7.35
T8	8.35	5.45	7.2
T9	8.25	5.35	7.4
T10	8.25	5.75	7
T11	8.25	5.55	7.2
T12	8.6	5.2	7.2
T13	8.25	5.2	7.55
T14	8.5	5.1	7.4
T15	8.5	5.4	7.1
T16	7.7	5.3	8
T17	7.7	5.55	7.75
T18	7.05	5.55	8.4
T19	7.75	5.55	7.7
T20	7.45	6.25	7.3
T21	7	6	8

Table (2) Results of applying the RR algorithm to the load matrix shown in Table (1)

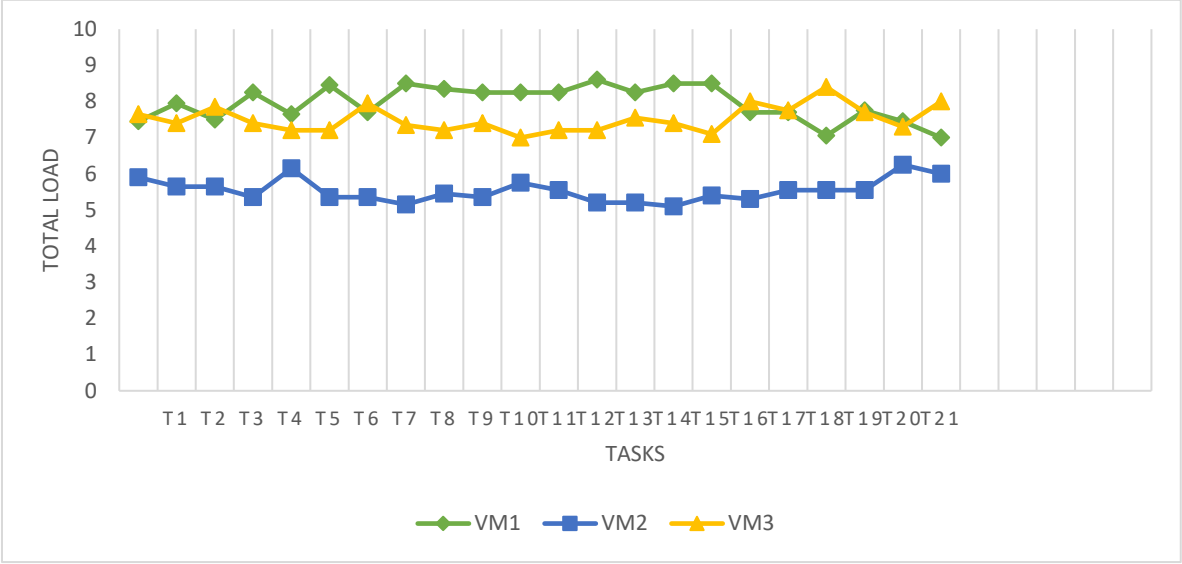
We note from Table (2) that the percentage of the total load for each virtual machine is different from the other, and by applying the relationship (8) to calculate the standard deviation, we find that:

$$Average(TL) = \frac{7+6+8}{3} = 7$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(TL(VM_j) - Average(TL))^2} = \sqrt{\frac{(7-7)^2 + (6-7)^2 + (8-7)^2}{3}} = 0.8164$$

Thus, the load balancing process was done, but with a standard deviation of the loads from the arithmetic mean value, and therefore this result is not the best result that can be reached and it did not achieve the best possible utilization of the available resources because this algorithm did not take into account the load caused by the task on the virtual machine during the allocation process mission.

The following figure shows the total load on each virtual machine during the task allocation process using the Round Robin algorithm, where we notice the differences in the load on each virtual machine, where there are virtual machines with a high load and others with a low load, and this is the reason for the increase in the value of the standard deviation.



The total load on each virtual machine during the allocation process shown in Table (2)

To apply the proposed approach of the load balancing algorithm to the matrix shown in Table (1), we first rearrange the tasks according to the virtual machines available for each task from lowest to highest, so we get the ordered matrix shown in Table (3):

	VM1	VM2	VM3
T2	0.45	X	0.55
T5	0.2	0.8	X
T6	0.75	X	0.25
T10	X	0.6	0.4
T11	X	0.2	0.8
T12	0.65	0.35	X
T13	0.35	X	0.65
T15	X	0.7	0.3
T17	X	0.75	0.25
T18	0.65	X	0.35
T19	0.3	X	0.7
T1	0.5	0.25	0.25
T3	0.25	0.3	0.45
T4	0.6	0.2	0.2
T7	0.2	0.2	0.6
T8	0.15	0.7	0.15
T9	0.1	0.1	0.8
T14	0.75	0.1	0.15
T16	0.8	0.1	0.1
T20	0.3	0.3	0.4
T21	0.45	0.25	0.3
Total Load (TL)	**7.45**	**5.9**	**7.65**

Table (3) Arrangement of tasks according to the virtual machines available for each task, from lowest to highest

Equation (5) is applied to the data of Table (3) to calculate the total load TL(VMj) where j=[1..4] on each virtual machine:

$$TL\ (VM1)=\sum_{i=1}^{21} Li1= 7.45$$

$$TL\ (VM2)=\sum_{i=1}^{21} Li2= 5.9$$

$$TL\ (VM3)=\sum_{i=1}^{21} Li3= 7.65$$

The virtual machine with the lowest overall overhead is chosen from among the available virtual machines to assign the task:

$$Min(TL(VM1),TL(VM3)) = TL(VM1)$$

The total load of the VM2 to which the task is allocated is updated according to equation (6):

$$TL\ (VM1) =7.45+(1-0.45)=8$$

Update the total load of the remaining virtual machines that were available for the task according to equation (7):

$$TL\ (VM3) =7.65-0.55=7.1$$

The overall load remains the same for virtual machines that were not available for the task:

$$. TL\ (VM2) =5.9$$

Table (4) shows the results of applying the previous steps to all tasks within the load matrix shown in Table (3). Assignment process for a new task.

	VM1	VM2	VM3
TL	7.45	5.9	7.65
T2	8	5.9	7.1
T5	7.8	6.1	7.1
T6	7.05	6.1	7.85
T10	7.05	6.5	7.45
T11	7.05	7.3	6.65
T12	7.4	6.95	6.65
T13	7.05	6.95	7
T15	7.05	7.25	6.7
T17	7.05	6.5	7.45
T18	7.4	6.5	7.1
T19	7.1	6.5	7.4
T1	6.6	7.25	7.15
T3	7.35	6.95	6.7
T4	6.75	6.75	7.5
T7	7.55	6.55	6.9
T8	7.4	6.85	6.75
T9	7.3	6.75	6.95
T14	6.55	7.65	6.8
T16	6.75	7.55	6.7
T20	6.45	7.25	7.3
T21	7	7	7

Table (4) Results of applying the proposed approach to the arranged load matrix shown in Table (3)

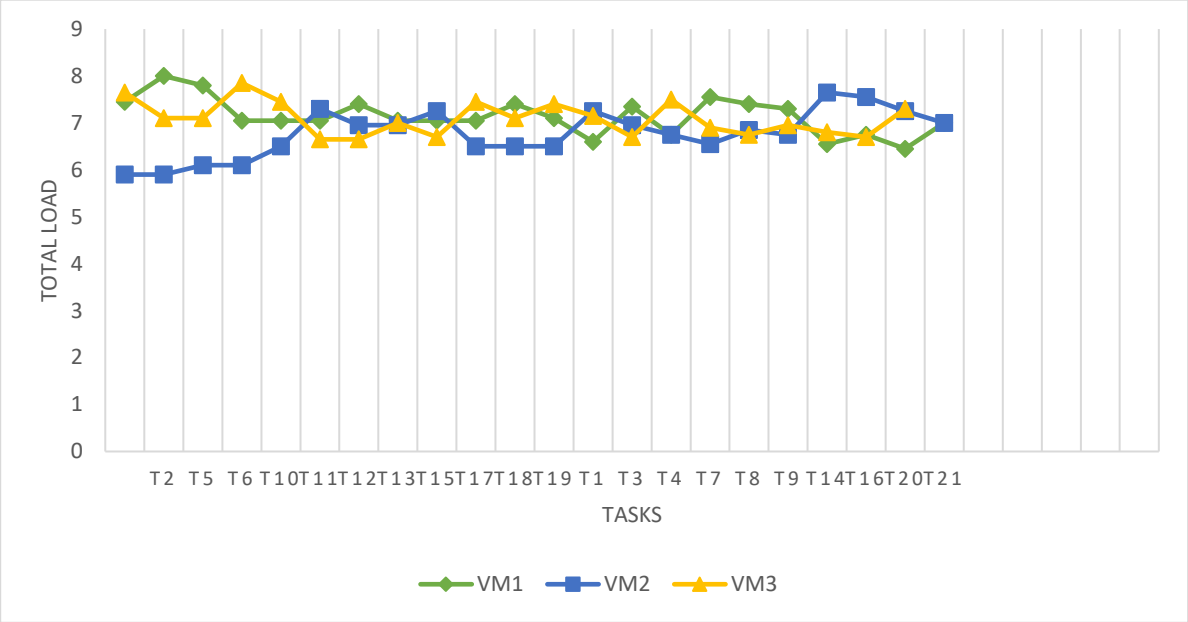
We note from Table (4) that the total load on each virtual machine after allocating all the tasks is 7, and each virtual machine is allocated 7 tasks, taking into account the physical resources of the virtual machine and the size of the task, and by applying the relationship (8) to calculate the standard deviation, we find that:

$$Average(TL) = \frac{7+7+7}{3} = 7$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(TL(VM_j) - Average(TL))^2} = \sqrt{\frac{(7-7)^2 + (7-7)^2 + (7-7)^2}{3}} = 0$$

We note that the value of the standard deviation $\sigma = 0$ is the best result that can be reached, as the load on each virtual machine after the completion of allocating all the tasks is the same, and this case represents the best possible utilization of the available resources because the tasks were distributed fairly on the virtual machines and taking into account the resources assigned to each virtual machine and the current load on each of them during the allocation process.

The following figure shows the total load on each virtual machine during the process of allocating tasks using the proposed approach, where we notice a large difference in the percentage of the load on each virtual machine before the start of the allocation process, then the loads on each virtual machine become very close to each other when starting to allocate the first task until they are equal in The end of the allocation process and the standard deviation value.



The total load on each virtual machine during the task allocation process shown in Table (4)

We note from the first scenario that the proposed approach achieves optimal load balancing that takes into account the capacity and resources of the virtual machine. We also note that the load ratio on each virtual machine is equal to the arithmetic mean value of all loads on all virtual machines, and therefore the standard deviation is equal to zero, which is the best result that can be reached. While the Round Robin algorithm, the load ratio between the virtual machines deviates from the arithmetic mean of the load, so the load balancing is done, but not in an optimal way that takes into account the resources allocated to each virtual machine.

3) In the second scenario, two tasks will be added to the load matrix shown in Joule (1), so that the number of tasks becomes not a multiple of the number of virtual machines, so we get the new matrix shown in Table (5):

	VM1	VM2	VM3
T1	0.5	0.25	0.25
T2	0.45	X	0.55
T3	0.25	0.3	0.45
T4	0.6	0.2	0.2
T5	0.2	0.8	X
T6	0.75	X	0.25

T7	0.2	0.2	0.6
T8	0.15	0.7	0.15
T9	0.1	0.1	0.8
T10	X	0.6	0.4
T11	X	0.2	0.8
T12	0.65	0.35	X
T13	0.35	X	0.65
T14	0.75	0.1	0.15
T15	X	0.7	0.3
T16	0.8	0.1	0.1
T17	X	0.75	0.25
T18	0.65	X	0.35
T19	0.3	X	0.7
T20	0.3	0.3	0.4
T21	0.45	0.25	0.3
T22	0.5	0.25	0.25
T23	0.75	X	0.25
Total Load (TL)	**8.7**	**6.15**	**8.15**

Table (5) Matrix loads with a number of tasks that are not multiples of the number of virtual machines

Table (6) shows the results of applying the Round Robin algorithm to the load matrix shown in Table (5), where the boxes in bold indicate the default machine to which the task has been allocated, and the values in the table show the change in the percentage of the load on each virtual machine at each assignment process New.

	VM1	VM2	VM3
TL	8.7	6.15	8.15
T1	9.2	5.9	7.9
T2	8.75	5.9	8.35
T3	9.5	5.6	7.9
T4	8.9	6.4	7.7
T5	9.7	5.6	7.7
T6	8.95	5.6	8.45
T7	9.75	5.4	7.85
T8	9.6	5.7	7.7
T9	9.5	5.6	7.9
T10	9.5	6	7.5
T11	9.5	5.8	7.7
T12	9.85	5.45	7.7
T13	9.5	5.45	8.05
T14	9.75	5.35	7.9
T15	9.75	5.65	7.6
T16	8.95	5.55	8.5

T17	8.95	5.8	8.25
T18	8.3	5.8	8.9
T19	9	5.8	8.2
T20	8.7	6.5	7.8
T21	8.25	6.25	8.5
T22	8.75	6	8.25
T23	8	6	9

Table (6) Results of applying the RR algorithm to the load matrix shown in Table (5)

We note from Table (6) that the percentage of the total load for each virtual machine is different from the other, and by applying the relationship (8) to calculate the standard deviation, we find that:

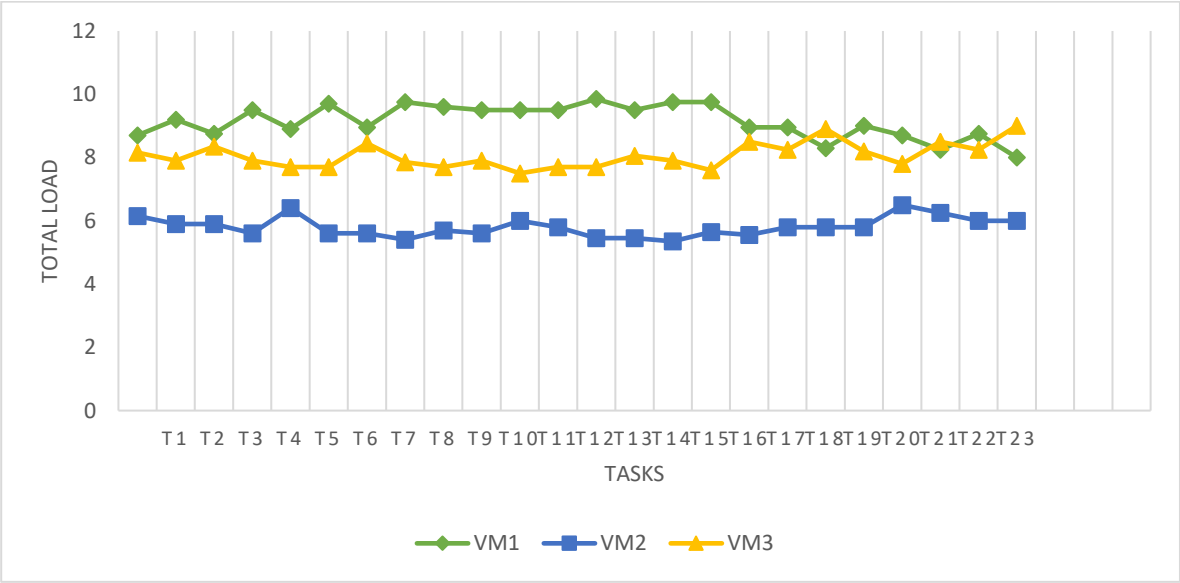
$$Average(TL) = \frac{8+6+9}{3} = 7.666$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(TL(VM_j) - Average(TL))^2}$$

$$\sigma = \sqrt{\frac{(8-7.666)^2 + (6-7.666)^2 + (9-7.666)^2}{3}} = 1.2472$$

Thus, the load balancing process was done, but with a large standard deviation, and therefore the tasks were not distributed fairly to the virtual machines, taking into account the current load on the virtual machines, and therefore did not achieve the best possible utilization of the available resources.

The following figure shows the total load on each virtual machine during the process of allocating tasks using the Round Robin algorithm. large in the standard deviation value.



The total load on each virtual machine during the task allocation process shown in Table (6)

Table (7) shows the results of applying the proposed approach to the load matrix shown in Table (5) after rearranging the tasks according to the virtual machines available for each task from the lowest to the highest.

	VM1	VM2	VM3
TL	8.7	6.15	8.15
T2	8.25	6.15	8.6

T5	8.05	6.35	8.6
T6	8.3	6.35	8.35
T10	8.3	6.75	7.95
T11	8.3	7.55	7.15
T12	7.65	8.2	7.15
T13	7.3	8.2	7.5
T15	7.3	7.5	8.2
T17	7.3	7.75	7.95
T18	7.65	7.75	7.6
T19	7.35	7.75	7.9
T23	7.6	7.75	7.65
T1	8.1	7.5	7.4
T3	7.85	7.2	7.95
T4	7.25	8	7.75
T7	8.05	7.8	7.15
T8	7.9	7.1	8
T9	7.8	8	7.2
T14	7.05	7.9	8.05
T16	7.25	7.8	7.95
T20	7.95	7.5	7.55
T21	7.5	8.25	7.25
T22	7	8	8

Table (7) The results of applying the proposed approach to the load matrix shown in Table (5) after their arrangement

Applying equation (12) to calculate the standard deviation, we find that:

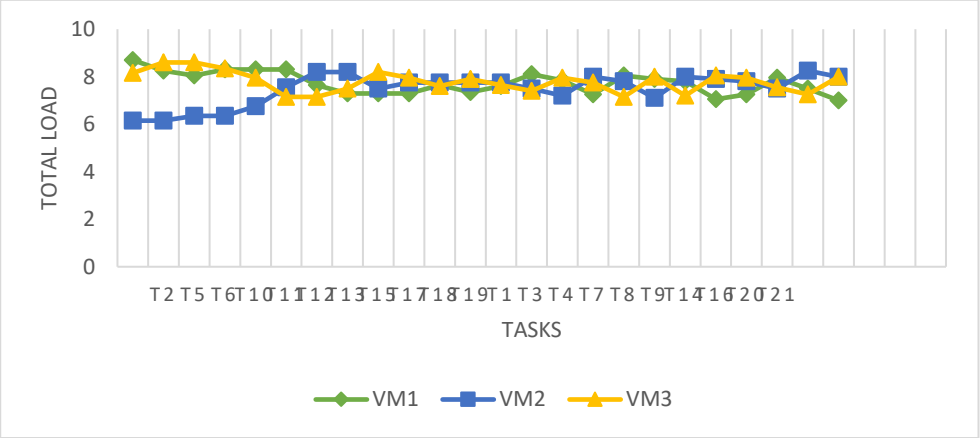
$$Average(TL) = \frac{7+8+8}{3} = 7.666$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(TL(VM_j) - Average(TL))^2}$$

$$\sigma = \sqrt{\frac{(7-7.666)^2 + (8-7.666)^2 + (8-7.666)^2}{3}} = 0.4714$$

We note from Table (7) that the load on the second and third virtual machines is greater than the load on the first virtual machine, and this is due to the fact that the second and third virtual machines have been assigned 8 tasks each, while the first virtual machine has been assigned 7 tasks, meaning that the two tasks that have been added One of them was assigned to the second virtual machine and one to the third virtual machine and this is due to a small standard deviation from the mean value.

The following figure shows the total load on each virtual machine during the process of allocating tasks using the proposed approach, where we notice a large difference in the percentage of the load on each virtual machine before the start of the allocation process, then the loads on each virtual machine become very close to each other until all tasks are allocated.



The total load on each virtual machine during the task allocation process shown in Table (7)

We note from the second scenario that the proposed approach is superior to the Round Robin algorithm in the process of load balancing, as the algorithm presented a bad load balance during task allocation and after the end of the allocation process caused a large standard deviation value compared to the proposed approach in which the load between virtual machines was very close because of the standard deviation value Simple.

We note from the first and second scenarios that if the number of tasks is a multiple of the number of virtual machines, then the proposed approach will achieve load balancing that is considered the best for the available resources and a standard deviation value equal to zero, but if the number of tasks is not a multiple of the number of virtual machines, the proposed approach remains superior to the Round algorithm Robin, although it causes a standard deviation value that is considered small compared to the standard deviation of the Round Robin algorithm, and also during the task allocation process using the proposed approach, the virtual machine loads were very close to each other, while in the Round Robin algorithm, there were large differences in the loads that led to the presence of virtual machines In the case of high load and the other in the case of low load, and therefore it may cause a bottleneck problem in the virtual machines with a high load, as well as some tasks that are allocated to the virtual machines with a high load may be dropped as a result of insufficient resources in them, while the proposed approach does not cause a fall In these problems because it takes into account the current load on the virtual machine and the load that the new task places on it.

7. Conclusions:

In this paper, we developed a probabilistic algorithm for load balancing in cloud computing platforms, within a heterogeneous environment in terms of the resources allocated to virtual machines. At the beginning of the research, a study was conducted that takes into account the resources allocated to each virtual machine and employing them in generating the load matrix. The results showed that the proposed algorithm provides effective and optimal load balancing compared to the load balancing results of the Round Robin algorithm. Load matrices were randomly generated and theoretical calculations were done to obtain the results. In the future, it would be better to simulate a realistic scenario and obtain the results and compare them with other algorithms. In this paper, it is considered that the access of tasks to the cloud data center is in batches, it will be more realistic if the access of tasks to the cloud data center is considered random [10][11][12], and the current approach is developed in a way that considers the random access of tasks.

References

[1] Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*, *7*(2), 421-426.

[2] Singh, P., Dutta, M., & Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowledge and Information Systems*, *52*(1), 1-51.

[3] Panda, S. K., & Jana, P. K. (2018). Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment. *Information Systems Frontiers*, *20*(2), 373-399.

[4] Chalack, V. A., Razavi, S. N., & Gudakahriz, S. J. (2017). Resource allocation in cloud environment using approaches based particle swarm optimization. *International Journal of Computer Applications Technology and Research*, *6*(2), 87-90.

[5] Gao, R., & Wu, J. (2015). Dynamic load balancing strategy for cloud computing with ant colony optimization. *Future Internet*, *7*(4), 465-483.

[6] Dhari, A., & Arif, K. I. (2017). An efficient load balancing scheme for cloud computing. *Indian Journal of Science and Technology*, *10*(11), 1-8.

[7] Phi, N. X., Tin, C. T., Thu, L. N. K., & Hung, T. C. (2018). Proposed load balancing algorithm to reduce response time and processing time on cloud computing. *Int. J. Comput. Netw. Commun*, *10*(3), 87-98.

[8] Chien, N. K., Son, N. H., & Loc, H. D. (2016, January). Load balancing algorithm based on estimating finish time of services in cloud computing. In *2016 18th International Conference on Advanced Communication Technology (ICACT)* (pp. 228-233). IEEE.

[9] Stephen, A., Shanthan, B. H., & Ravindran, D. (2018). Enhanced round Robin algorithm for cloud computing. *Int J Sci Res Comput Sci Appl Manag Stud*, *7*(4), 1-5.

[10] Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*, *7*(2), 421-426.

[11] Singh, P., Dutta, M., & Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowledge and Information Systems*, *52*(1), 1-51.

[12] Panda, S. K., & Jana, P. K. (2018). Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment. *Information Systems Frontiers*, *20*(2), 373-399.

[13] Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, *49*(4), 764-766.