Journal of Education for Pure Science- University of Thi-Qar
Vol.10, No.2 (June, 2020)
Website: jceps.utq.edu.iq *Email: jceps@eps.utq.edu.iq*

DOI: http://doi.org/10.32792/utq.jceps.10.02.07

A hybrid approach for load balancing in cloud computing

Ghsuoon Badr Roomi1 Dr. Khaldun. I.Arif2

1,2 Computer Science Department, College of Technical Thi-Qar, Southern Technical University,

Thi - Qar, Iraq

Received 13/10/2019 Accepted 8/03/2020 Published 30/11/2020

Abstract:
Cloud computing is one of the most promising technical developments in recent days. It emerged as a dominant and transformational paradigm. Cloud computing can be considered a vital form of information technology that allows the delivery of services to users via the Internet upon request from the user and based on immediate payment. One of the main challenges and important fields for research in the cloud computing environment is load balancing. Load balancing has become an important point for stability and good system performance. Therefore, the main goal is to establish an effective load balancing algorithm for task scheduling. In this paper, we introduce the Hybrid Algorithm for Load Balancing (HALB), which aims to balance effective loading among virtual machines by balancing the percentage of data usage from RAM in each VM. As the results of the percentages were close, and all percentages did not reach the condition of overload. The proposed hybrid algorithm also reduces average waiting time and turnaround time. As a mechanism of work of our hybrid algorithm relies on two types of scheduling algorithms, one dependent on the other, namely the lottery algorithm and the Shortest Job First algorithm. A specific mechanism has been implemented to allocate tasks resulting from scheduling each algorithm separately, when calculating the total size of data tasks in each VM, the results showed that the volume of data allocated within the VM. Data sizes converge across all VMs.
Keywords: Cloud computing, Load Balancing, Task Scheduling, Lottery, Shortest Job First.

1. Introduction:

Cloud Computing is still a source of confusion to many of us already. Cloud computing is a parallel and distributed computing system. It contains groups of inter-connected, and virtualized physical machines that are provided dynamically. Cloud computing allows the sharing of resources, software and information. To use cloud computing, the Internet must be available. Cloud computing has been defined as a "technology" that relies on the transfer of processing and computer storage space to the so-called cloud, a data center accessed by the network, thereby transforming IT programs from products to services, which contributes to the removal of maintenance and system development problems of the companies used for them, The National Institute of Standards and Technology (NIST) described cloud computing as a paradigm that enables ubiquitous, convenient, and network can be accessed to a shared pool of cloud computing resources that can be provisioned quickly [1]. Cloud computing spreading in our lives as Email. For instance, web mail services such as Google's Gmail, Microsoft's Hotmail and Yahoo Mail, is cloud-typed services since

customers store their data (emails) on the providers' data centers. Facebook is also a cloud-typed service and so on. You can access your email at Anywhere and any time via a provision Internet connection, users should be pay-as-he use based on the utilization of cloud services, like to TV services, electricity, and so on [2]. cloud computing providers and presents developers of computer applications and users at the same time an abstract destination that facilitates and ignores many internal details and processes. Here. one of the largest companies offering cloud computing service is Amazon, which is considered the founder of the cloud [3]. Cloud computing idea shown in Figure 1.

Figure 1: Cloud Computing

2. Research Problem:

Cloud computing has emerged as a new and essential field used to increase the performance of distributed computing systems. Where data software and equipment are available online. It is a set of heterogeneous systems in different regions. Data centers consist of a large number of physical devices that contain virtual machines (VMs). Each VM has a particular configuration of processing power, bandwidth, RAM, and storage based on the capacity of each device. Tasks are assigned to each device. This large number of tasks affects the performance of virtual machines, resulting in a load imbalance problem. Load balancing between virtual machines must be equal. Load balancing is a complex condition due to dynamic nature and heterogeneous sources. For this reason, load balancing algorithms distributed loads on different physical devices and VM to maximize resource utilization and reduce turnaround time, the completion time of tasks, waiting time and usage data of VM RAM.

3. Research objective:

The primary purpose of the research is to propose an algorithm whose advantage is that it can find the balance between the sources and the tasks coming from the user based on certain measurements by scheduling a large number of tasks. To achieve the core objectives, we have implemented the following purposes:

1. To design A Hybrid Algorithm for Load Balancing (HALD). That can balance the load between the virtual machines and the distribution of tasks between them, as scheduled by an efficient scheduling algorithm.
2. To reduce load balancing metrics such as turnaround time, completion time, and reduce the percentage of usage data of VM ram used for each virtual machine.

3. To evaluate the results of the) HALD (algorithm and compare it with the algorithms used as tools in the hybrid algorithm using the cloudim3.0.3 toolkit and the NetBeans11.2 IDE program.

4. Related work:

In 2016 Sidana, Shubham, et al. [4]. In this paper, the authors try to balance the load by arranging virtual machines based on processing power and arranging cloudlets according to their length, i.e. the number of instructions in the cloud. After that the list of virtual devices and cloudlets is sent to broker for the allocation. Broker allocates through mid-point algorithm and divides the VM list and the cloud list so that at least one cloudlet or virtual machine remains in the list then these resources are allocated. By the proposed algorithm, they assumed that the jobs are in the queue and they know the length, that is, the number of instructions in the request. Load Balancing Algorithm aims to reduce the load on resources. To accomplish this, all the virtual machines are ranked in order of MIPS execution speed (million instructions per second). After arranging the machines, sorting of cloudlets is performed based on their length (million instructions). Mid-point is taken of those sorted cloudlets list and sorted virtual machines list and then the divided cloudlet lists are mapped to the corresponding lists of virtual machines.

In 2017, Lagwal, Monika, and Neha Bhardwaj [5]. In this research, the researchers tried to balance the work-load by arranging VM on the basis of their processing power and arranging the cloudlets according to their Length i.e. number of instructions in the cloudlet. The list of VM and cloudlets is then submitted to broker for the allocation. Broker allocates through GA, with the help of Genetic Algorithm) GA (. Then, the broker assigns effective processes to clients on the basis of the cost and time effective manner. For this use GA with cloud-sim tool, used GA approach for better and efficient (time) results. In 2017, Dhari, Atyaf, and Arif, K [6]. In this work, researchers presented an algorithm to improve system performance, by balancing the load burden among virtual machines. They proposed a LBDA algorithm to balance the load between virtual machines in the data center to reduce completion time (Makespan) and response time. The LBDA mechanism relies on three phases in three phases; first, it calculates VM amplitude and VM load to classify states of VMs (under loaded VM, balanced VM, high balance VM, and overload). Second, they calculate the time the task takes to execute in each VM and finally, it makes a decision to distribute the tasks between the VMs based on the VM status and the time of the requested task. In 2019, Maheshwari, Khushboo, and Ved Kumar Gupta. [7] Researchers highlighted a method measure the performance of cloud scheduling approaches that are claimed to optimize the performance of the cloud computing. The resource scheduling and the workload balancing is the similar directional effort for optimizing the computational performance of the cloud infrastructures. Therefore, two popular and frequently used scheduling approaches i.e. round robin and first come first serve techniques are implemented with the help of cloudSim simulator. The round robin technique allocates a fixed amount of time for all the resources to a given job therefore this concept works as the time shared manner. Similarly, the FCFS technique allocates jobs according to their appearance or sequence therefore that technique is functions according to the space shared manner. Additionally, the performance of both the approaches are measured and compared. In order to compare the performance of both the techniques the average processing time, average processing cost, average waiting time and the CPU utilization is computed using the simulation trace.

4.1. Comparing the proposed hybrid algorithm with related work in table 1:

Num	Scientist name	Method name	Performance standards used	Result	Number VM and Number Tasks	balancing Data and Time in VM	
						Data	Time
1	Shubham Sidana, Neha Tiwari,Anurag Gupta and Inall Singh Kushwaha	NBST Algorithm: load A balancing algorithm in cloud computing	execution time	45ms	Number VM=8 Number Tasks =16	no	high
2	Lagwal, and Monika, Neha Bhardwaj	CLB Algorithm: Cloud Load balancing	execution time Finish time	34.736ms 16.7ms	Number VM 9 Number Tasks 12	no	high
3	Dhari, Atyaf, And KHALDun. Arif	LBDA :Load Balancing Decision Algorithm	Average Makespan, Mean of Average Response Total Execution Time	59ms 54ms 223.33ms	Number VM=2 Number Task=4	no	high
4	The proposed hybrid algorithm	HALB: Hybrid Algorithm for Load Balancing	average turnaround time average waiting time total data size of 25 VM Usage VM Ram	99.2ms 49.316ms 121610.2MB 39.59MB	Number VM=25 Number Task =100	yes	low

a) . Load Balancing in Cloud Computing:

Load balancing has been considered one of the most critical aspects of cloud computing in recent times. Which efficiently distribute the workloads across a group of backend servers. Modern Web sites serve millions of concurrent requests from users and return the correct information as in the form of images, text, and video, all in a fast and reliable method. To meet the requirements, cloud computing usually needs more

servers for facilitating the services, which is cost-effective [8]. An increase in the number of users around the world has resulted in a large number of requests at a rapid rate. Researchers around the world have designed many algorithms to carry out the client's request at distributed cloud servers [9].

b) 6. The Proposed Methods:

We are discussing the problem of load balancing in cloud computing, where we focused on how to access the optimal load balancing by arranging tasks that come from the user by scheduling algorithms to distribute on virtual machines. The virtual machine is the primary unit responsible for a task. These tasks require a special resource, and scheduling algorithm that evaluates system parameters such as task completion time and virtual machine state. At this point, the virtual machine resource details are configured in the system. Then all the details of the tasks are entered. Hybrid algorithm for load balancing resulting from combining the lottery and shortest job first algorithms.

(1) 6.2 The Steps Hybrid Algorithm for Load Balancing (HALB)

general steps of the Hybrid Algorithm for Load Balancing
Input: tasks and number of VM
Output: statistical and results of the implementation for each VM
Begin Step 1: initializing the number of tasks and VM. Step 2: check if the value of task > 20000 and < 29999 Then sort the tasks in array (SJF array). Step 3: the system begins Appling lottery algorithm on the tasks that between (30000-40000), the Results values stored in array (lottery array). Step 4: the system starts creating new array (array all) by mixing between the (SJF array).and (Lottery array), the mixing is done by selecting one Task form each array at time. Step 5: the Step of assigning tasks from (Array all) among VM's Step 6: apply the SJF for each VM and extract the results And the statistical that shows the load balance the among VMs. End

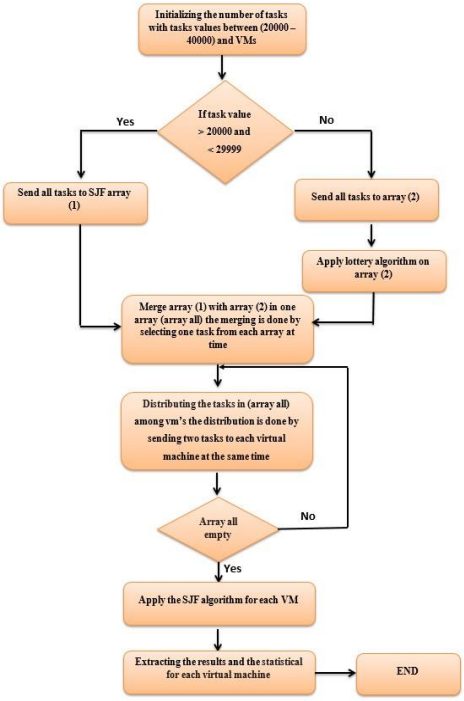

Figure 2: Algorithm (HALB)

7. EXPERIMENTS:

4.2 7.1. Introduction:

This section includes the results of the proposed hybrid Algorithm. We have been keen in our proposed hybrid algorithm to provide an effective load balance between virtual machines using scheduling algorithms that support one another; the lottery algorithm has a fair scheduling feature for tasks and the random selection of tasks with winning tickets. The SJF algorithm features that it schedules the shortest tasks first, which gives a better Average Turnaround Time and less the average waiting time for the tasks. Which gave this proposed hybrid algorithm the power with which it was able to achieve VM's load balance and was able to reduce task waiting time and give Average Turnaround Time in the best possible state. We proposed a special mechanism for the distribution of tasks on virtual machines, which consequently produced load balancing on virtual machines equally among virtual machines.

4.3 7.2. Experiments Design and Setup:

The proposed hybrid algorithm was applied to four experiments. The number of tasks and the number of virtual machines differ between experiments. This is to demonstrate the efficiency and ability of the algorithm to balance loads between virtual machines. The hybrid algorithm is able to balance the load of virtual machines with different number of tasks in each experiment. We explain the first experiment in this section where the number of tasks that have been entered is 35 tasks and the number of VM 5. The tasks data entered was between 20,000 to 40,000 MI (Million instruction). The calculations made in the first experiment for all virtual machines are all the following:

1. Turnaround time: (TAT) means response time the amount of time it takes to complete the process or fulfill the request. Response time is the time interval from the process entering the VM to the end time.

TA [i] = CT [i] - AT [i] ..…… (1) where CT is the full time for the task, TA is the time for the task to arrive. To calculate the average response time, divide the total response time for all tasks by the number of tasks.

2. The process waiting time is the amount of time that has been waiting for it in the ready queue.
Waiting time: The time difference between response time and burst time.

WT [i] = TA [i] -BT [i] ……. (2)

Where BT is the time of the mission explosion.
To compute the average waiting time, we divide the summation of the waiting time of all tasks on a number of tasks.
Average wait time, total data size, and VM Ram usage on all virtual machines, in Table 1.

3. The total data size is calculated for each VM, whereby the total data volume is calculated for all tasks within the VM. Next, we will compute the percentage of data usage from the RAM in each virtual machine.

4.4 7.3. Experiments Results:

In this section, we present the results obtained from experiments. The proposed hybrid algorithm was applied to four experiments. The number of tasks and the number of virtual machines differ between experiments. And this is to demonstrate the efficiency and ability of the algorithm to balance loads between virtual machines. We will discuss the results of the first experiment. The number of tasks entered was 35 and the number of VM was 5, and the lengths of the tasks entered were between 20,000 to 40,000 MI. Tasks are divided according to their lengths ranging from 20,000 to 29999 MI and the other part with lengths ranging from 30,000 to 40 000 MI, which will be the first to be handled by the lottery algorithm. After this stage comes the Array_all creation phase whose elements consist of an element that takes from Array SJF and an element that takes from Array_lot. The size of this array is all of the tasks within the proposed hybrid algorithm. As for the assignment mechanism to virtual machines, we take the first two tasks from Array_all and put them in the VM1 array. The following two tasks are assigned to the VM2 array and so on until accessing the VM5 array and repeating this process until each VM array contains seven tasks.

Table2: shows results of all VM and tasks assigned to each VM.

Task / VM	Task1	Task2	Task3	Task4	Task5	Task6	Task7
VM1	25400	31475	21007	37906	24619	39638	27088
VM2	21755	32471	22917	31677	35912	32669	22926
VM3	26447	38411	20717	32354	28362	35854	25543
VM4	22682	30476	23525	35484	39137	32587	29472
VM5	27844	37362	24320	37678	27696	34807	37260

At this point, the virtual machines start to work were at this point the algorithm SJF that was applied to the tasks inside the VM has been used run of tasks starts in VM. We will show the results of each VM1 implementation separately.

Table3: shows results arrival time, waiting time, turnaround, complete time, burst time for vm1.

Task id	Arrival time	Burst time	Complete time	Turnaround time	Waiting time
1	0	68	175	175	107
2	6	84	329	323	239
3	9	14	28	19	5
4	10	5	15	5	0
5	14	47	116	102	55
6	18	41	69	51	10
7	28	70	245	217	147

Table 4: Shows Results of all calculations for each VM separately

VM Number	Average Turnaround Time	Average Waiting Time	Total Data Size	Usage of VM Ram
VM1	127.42	80.42	207133mb	67.42%
VM2	142.28	94.85	200327mb	65.21%
VM3	198.57	136.14	207688mb	67.66%
VM4	123.0	79.28	213363mb	69.45%
VM5	77.28	44.57	226967mb	73.88%

4.1 7.4. Results and Discussion:

We reach the stage of evaluating the results based on the comparison of the hybrid algorithm for the load balancing (HALD) with the algorithms that contribute to this hybrid algorithm, which is the SJF algorithm and the lottery algorithm are important tools in the mixed algorithm. The algorithms were implemented using the CloudSim 3.0.3 toolkit and NetBeans IDE11.2 simulating the cloud environment and simulation results and measuring them by units per second for Waiting Time and turnaround time and (MB) for data. The results of the hybrid algorithm were compared with these two algorithms, and we did not compare with previous research to demonstrate the efficiency and effectiveness of the hybrid algorithm in balancing the loads between VM. We explain this by comparing the algorithms, each algorithm separately. Table (5) shows the results of the average response time, where the HALD achieved better results and better than other algorithms, where when experimenting with the SJF and lottery algorithms with the same number of tasks which is 35 and 5 VM, average response time HALD algorithm was lower compared to other algorithms. We also notice that the average turnaround time for the SJF algorithm at the beginning of the table for virtual machines at the front of the table is less than the results for the hybrid algorithm (HALD). Depending on how the SJF algorithm is scheduled tasks, the shortest task is first. This explains why results at the beginning of the table are less than the hybrid algorithm.

Table 5: Shows a comparison results Average Turnaround Time among HALB, SJF and Lottery

Average Turnaround Time			
VM id	HALB	SJF	Lottery
VM1	127.42	96.43	140
VM2	142.28	107.3	163.71
VM3	198.57	255.14	202.40
VM4	123	143.86	156
VM5	77.28	98.29	127.14

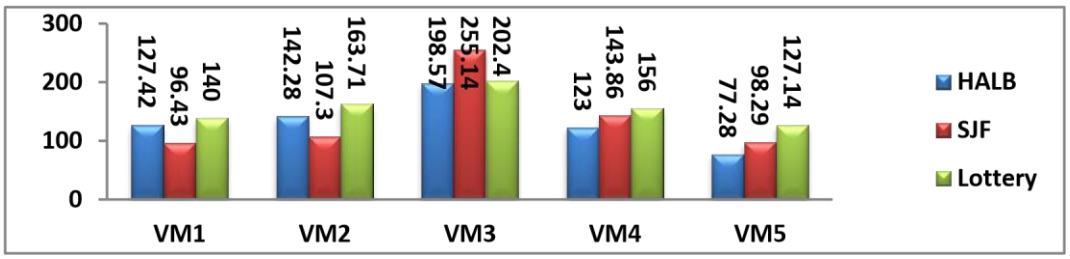

Figure 3: Comparison of Average Turnaround Time among HALB, SJF and Lottery

Table 6: Shows a comparison results Average Waiting Time among HALB, SJF and Lottery.

Average Waiting Time			
VM id	HALB	SJF	Lottery
VM1	80.42	63.57	99.3
VM2	94.85	63.43	116
VM3	136.14	190.29	184.5
VM4	79.28	92.28	110
VM5	44.57	60.57	84.3

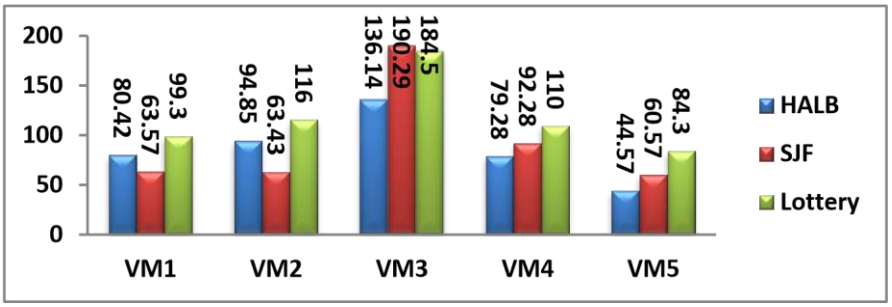

Figure 4: Comparison of Average Waiting Time among HALB, SJF and Lottery

Table 7: Shows a comparison results Total Data Size in VM among HALB, SJF and Lottery.

Total Data Size in VM			
VM id	**HALB**	**SJF**	**Lottery**
VM1	207133 MB	155529 MB	230578 MB
VM2	200327 MB	181113 MB	250660 MB
VM3	207688 MB	211660 MB	216989 MB
VM4	213363 MB	239784 MB	246989 MB
VM5	226967 MB	267392 MB	239889 MB

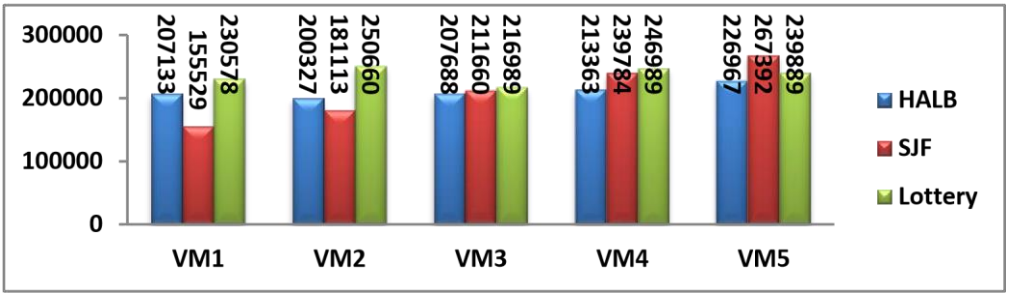

Figure 5: Comparison of Total Data Size in VM among HALB, SJF and Lottery

Table 8: Shows a comparison resulted Usage of VM Ram among HALB, SJF and Lottery.

Usage of VM Ram			
VM id	**HALB**	**SJF**	**Lottery**
VM1	67.42%	50.63%	75.05%
VM2	65.21%	58.97%	81.59%
VM3	67.60%	68.89%	70.63%
VM4	69.45%	78.05%	80.40%
VM5	73.88%	87.04%	78.09%

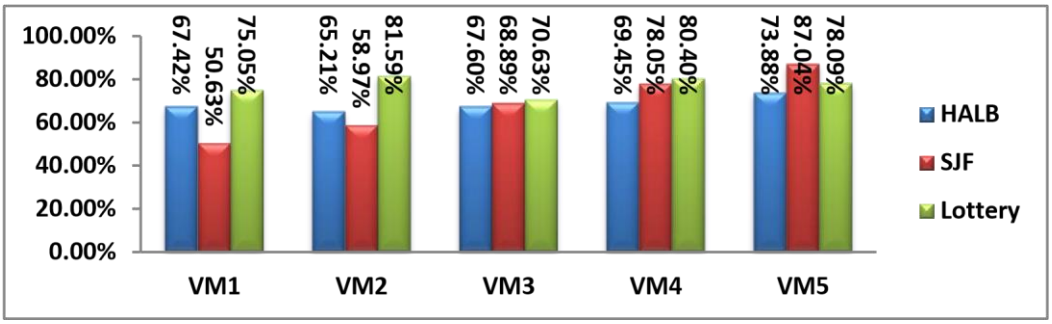

Figure 6: Comparison of Usage of VM Ram among HALB, SJF and Lottery

The results showed our proposed algorithm (HALB) better results as well as less result than the compared algorithms. Where was (Mean of total Average Turnaround Time) and (Mean of total Average Waiting Time) is lower in all trials than the lottery algorithm and the SJF algorithm. In addition, data results for tasks are Average Total Data Size in VM and Average Usage of VM Ram for all experiments

(HALB) showed results less data than in the algorithms that were compared with them.

Table 8: Showing the results of comparing (HALB) with the lottery algorithm and algorithm (SJF) for the four experiments for all the calculations in the four experiments.

Exps.	No. of tasks and VMs	Mean of total Average Turnaround Time			Mean of total Average Waiting Time			Average Total Data Size in VM			Average Usage of VM Ram		
		HALB	SJF	Lottery	HALB	SJF	Lottery	HALB	SJF	Lottery	HALB	SJF	Lottery
1	35Tasks in 5 VMs	133.71	140.20	157.85	87.052	94.028	118.82	211095.6	211095.6	237021	68.71%	71.72%	77.15%
2	100 Tasks in 25VMs	99.2	111.17	133.06	49.316	62.892	64.2	121610.2	129831.52	136383.56	39.59%	42.26%	45.09%
3	120 Tasks in 20VMs	123.85	123.88	138.77	76.271	86.296	89.82	167054.1	179956.15	188955.8	54.37%	58.57%	61.50%
4	180 Tasks in 30 VMs	123.6	140.8	175.5	75.84	88.56	95.23	174031.7	186121.4	198864.3	56.64%	60.58%	64.73%

4.2 8. Conclusions

The main goal of load alancing is tobalance the load equally among the virtual machines such that no virtual machines will be overloaded or under loaded. This thesis consists of a comprehensive overview related to load balancing algorithm in cloud computing and resolve the issue related to the load balancing. We evaluated the proposed HALB performance with current algorithms like SJF and Lottery. The proposed algorithm (HALD) was implemented, and the algorithms were compared with it. Comparisons were made with the same environment configurations in the cloudim3.0.3 and NetBeans IDE 11.2 toolkit. The evaluation was based on several trials. The number of tasks and the number of virtual machines vary from experience to experience. We increased the number of tasks and the number of VM between experiments and the proposed algorithm achieved load balance, despite the increase in the number of tasks from one experiment to another. The main important conclusion is:

1. This research has introduced a hybrid algorithm was, combining the lottery algorithm with the algorithm SJF. The idea is to combine these two algorithms because of their characteristics that support each other and the best result was achieved as a result of scheduling tasks when they were combined.
2. Lottery algorithm feature is a fair tasks scheduling where tickets are given for tasks and winning tickets are randomly selected. This random feature for choosing tasks avoids starvation. As for the introduction of an algorithm (SJF), it schedules shorter tasks, they implement first of their features, which reduces the average waiting time for tasks. The characteristics of each of the algorithms gave strength and efficiency to the hybrid algorithm with a load balance between the virtual machines (VM's) in all experiments.
3. A specific mechanism is proposed to distribute tasks to virtual machines. This includes tasks resulting from scheduling lottery algorithm and SJF algorithm and they are all placed in an array. After that, the tasks are distributed in a special way to the virtual machines. This proposed method yielded the best results for assigning tasks to virtual machines. This method produced the best load balancing.

4. A new method has been proposed. The idea behind this method is to set the task lengths between 20000 to 40000MI (million instructions). The task lengths have been divided from 20000 to 30000MI within the short tasks section 30000 to 400000MI within the long tasks section. This method produced an effective balance for the tasks entering the virtual machines. This led to a high performance efficiency to balance the data in the RAM inside the virtual machine.
5. The hybrid algorithm is designed and implemented to balance the data load in virtual machines, as well as speed up the response time to the tasks and reduce the average waiting time for the tasks. as the hybrid algorithm was balanced and the data ratios in all the experiments were close in all the virtual machines, and the state of the overload did not reach.
6. The average turnaround time and average wait time are lower than other algorithms in all trials. This parameter is important for the user and we made sure to meet the user's satisfaction.
7. In Hybrid Algorithm for Load Balancing HALB, we avoided starvation and wasted resource utilization while other comparison algorithms were suffered from it.
8. Experiments showed that all the results of the Hybrid Algorithm for Load Balancing (HALB) outperform other algorithms and return less the size of data in VM and the percentage usage of VM ram, the results are less than other algorithms.
9. We sought to make the load balance achieved by the hybrid algorithm for virtual machines be on two sides. The data side, where the tasks data that is sent to the virtual machines has been balanced. On the other hand, time gave the lowest average turnaround time and the lowest average waiting time and completion time.
10. The Hybrid Load Balancing Algorithm (HALB) is a combination of the Lottery Algorithm and the SJF Algorithm. It is a new method and no one has touched upon previously. This was a reason we did not find comparative research in order to be compared, we conducted many experiments and we combined previous methods of scheduling tasks but did not give satisfactory results and the results were below the required level. We have therefore evaluated the performance of the proposed HALB with current algorithms such as SJF and Lottery. The proposed algorithm (HALB) was implemented, and compared with algorithms. Comparisons were made with the same environment configurations in cloudim3.0.3 and NetBeans IDE 11.2 toolkit.

8. References:

[1] Muche, Elias Wondmagegn. "Hybrid intrusion detection system for private cloud: anintegrated approach." Master thesis in Computer Science, Bahir Dar University, January 2016.
[2] Deore, Shailesh Shivaji. Design and optimization of scheduling schemes for cloud computing. PhD thesis, Shri Jagdishprasad Jhabarmal Tibarewala University, Jeanery 2013.
[3] Ismail, E., & Alamri, F. Optimized Load Balancing based Task Scheduling in Cloud Environment. *International Journal of Computer Applications, Majan College International Conference, 35-38, 2014.*
[4] Sidana, Shubham, et al. "NBST algorithm: A load balancing algorithm in cloud computing." International Conference on Computing, Communication and Automation (ICCCA). IEEE, 2016.
[5] Lagwal, Monika, and Neha Bhardwaj. "Load balancing in cloud computing using genetic algorithm." *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2017.
[6] Dhari, Atyaf, and KHALDun I. Arif. "An efficient load balancing scheme for cloud computing." Indian Journal of Science and Technology 10.11 (2017): 1-8.

[7] Maheshwari, Khushboo, and Ved Kumar Gupta. "Load Balancing in VM in Cloud Computing Using CloudSim." *Proceedings of Recent Advances in Interdisciplinary Trends in Engineering & Applications (RAITEA)* (2019).

[8] Basu, Srijita, and Abhishek Anand. "Location Based Secured Task Scheduling in Cloud." *Information and Communication Technology for Intelligent Systems*. Springer, Singapore, 2019. 61-69.

[9] Kaur, Rajveer, and Supriya Kinger. "Analysis of job scheduling algorithms in cloud computing." *International Journal of Computer Trends and Technology (IJCTT)* 9.7 (2014): 379-386.

[10] Salazar, Maria Helena Mejia, and Tapasya Patki. "Lottery Scheduler for the Linux 2.6 Kernel." (2010).

[11] Mejía, María, Adriana Morales-Betancourt, and Tapasya Patki. "Lottery scheduler for the Linux kernel." Dyna 82.189 (2015): 216-225.

[12] Manuel, Jezreel Ian C., et al. "Fittest Job First Dynamic Round Robin (FJFDRR) scheduling algorithm using dual queue and arrival time factor: a comparison." IOP Conference Series:
Materials Science and Engineering. Vol. 482. No. 1. IOP Publishing, 2019.