Pertanika J. Sci. & Technol. 32 (1): 113 - 125 (2024)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Use of Enhanced Greedy Algorithm for Load Balancing in Cloud Computing

Hanaa Osman[1], Asma'a Yassin Hammo[1] and Abdulnasir Younus Ahmad[2]*

[1]*Computer Science and Mathematics, University of Mosul, 259 AL-Majmoaa St., Mosul, Nineveh, Iraq*
[2]*Education for Pure Science, University of Mosul, 259 AL-Majmoaa St., Mosul, Nineveh, Iraq*

ABSTRACT

Because of the Internet's phenomenal growth in recent years, computing resources are now more widely available. It led to the development of a new computing concept known as Cloud Computing, allowing users to share resources such as networks, servers, storage, applications, services, software, and data across multiple devices on demand for economical and fast. Load balancing is an important branch of cloud computing as it optimizes machine utilization by distributing tasks equally over resources. It occurs among physical hosts or Virtual Machines in a cloud environment. Round robin is a commonly used algorithm in load balancing. RR gives a time quantum for each task and is in circular order. It is noted that it suffers from many problems, such as the waste of time and the high cost. In the present study, the greedy algorithm was enhanced and implemented to allocate and schedule tasks that come to the cloud on Virtual Machines in balance. The task with the longest execution time is given to the virtual machine with the least load using an improved greedy algorithm. The outcomes demonstrate that the suggested algorithm outperformed round robin in makespan. Also, all Virtual Machines in the proposed algorithm finish their work simultaneously, whereas round robin is unbalanced.

Keywords: Cloud computing, cloudSim, greedy algorithm, load balancing, makespan, round robin, Virtual Machine

ARTICLE INFO

Article history:
Received: 25 October 2022
Accepted: 14 June 2023
Published: 24 November 2023

DOI: https://doi.org/10.47836/pjst.32.1.07

E-mail addresses:
hanaosman@uomosul.edu.iq (Hanaa Osman)
asmahammo@uomosul.edu.iq (Asma'a Yassin Hammo)
abdulnasir.younus@uomosul.edu.iq (Abdulnasir Younus Ahmad)
* Corresponding author

INTRODUCTION

Computing resources are now more widely available, allowing for the development of a new computing concept known as Cloud computing because of the Internet's phenomenal growth in recent years (Fatima et al., 2019). Cloud computing is a kind of Internet-based computing that enables

ISSN: 0128-7680
e-ISSN: 2231-8526

clients to share resources, software, and data across various devices on demand (Mishra et al., 2018). Generally, the cloud computing platform has three major issues: virtualization, distributed framework, and load balance (Kumar et al., 2020).

Virtualization technology improves cloud resource use by making various resources available online for customers to purchase on a pay-per-use basis. Physical servers (maybe even only one) can be set up and divided into numerous unconnected "virtual servers" (PS), each of which functions independently and appears to the user to be a single physical device. PSs can be moved in any direction and scaled up or down without causing harm to the end user because they are not physically tied (Fatima et al., 2019).

A distributed system is made up of numerous independent computers that communicate with one another via a computer network. Computers communicate to reach a common aim (Nerkar, 2012). A distributed cloud that connects several geographically dispersed and smaller data centers can be a compelling alternative to today's big, centralized data centers. A distributed cloud can reduce communication overheads, prices, and latency by providing adjacent processing and storage resources. Improved data locality can also help with privacy. Many smaller data centers are installed closer to consumers to complement or enhance the bigger mega-data centers under the distributed cloud deployment paradigm, and the smaller data centers are administered as a single pooled resource (Coady et al., 2015).

When certain Virtual Machines (VMs) are overloaded with processing duties while others are underloaded, the load must be balanced to achieve optimal machine utilization (Babu & Krishna, 2013; Kumar & Kumar, 2019). Typically, web traffic, application access, databases, and other entities with large loads can use load balancer software to support uninterrupted service to Clients.

Load balancing may occur among physical hosts or Virtual Machines in a cloudy medium (Kumar & Kumar, 2019).

There are two basic challenges with load balancing:

- Task allocation refers to the distribution of a fixed number of tasks over a large number of Physical Machines (PMs), followed by VMs concerning the PMs.
- VMs Relocation Management: The process of moving virtual machines from one PMs to another to increase the data center's resource consumption is called Migration (Kumar & Kumar, 2019).

Millions of users share cloud resources by submitting their computing tasks to the cloud system. The cloud computing environment faces a hurdle in scheduling these millions of tasks. The scheduling of cloud services is divided into two categories: user and system. Scheduling at the user level addresses issues arising from service supply between providers and customers. Within the data center, resource management is handled through system-level scheduling (Tawfeek et al., 2013).

Some of the most typical goals of adopting load balancing techniques are reducing waiting time, reducing the response time, increasing the utilization of resources, improving

reliability, increasing throughput, minimizing turn-around-time, and minimizing makespan (Dave & Maheta, 2014).

One of the most common load balancing algorithms used is round robin (RR). Although its technique is straightforward and convenient, it has several flaws, including time loss and a high cost. It is primarily due to the job being assigned to the incorrect virtual machine, which does not consider the task's size or the requirements of the virtual machine to which it is attached. This work aims to enhance and implement a greedy algorithm to allocate and schedule tasks to the cloud on balance. The algorithm is compared with what is called the RR algorithm.

Related Works

Caragiannis et al. (2011) investigate two scenarios: selfish load balancing and online load balancing. The researchers describe the impact of selfishness and greediness on load balancing. They show that anarchy and stability of selfish load balancing have been enhanced and tightened. They also constrain the greedy algorithm's competitiveness for online load balancing, where the goal is to reduce latency for all clients on servers. Babu and Krishna (2013) proposed an algorithm named honey bee behavior-inspired load balancing (HBB-LB). The proposed algorithm balances the priority of tasks on the machines with the waiting time. When compared to existing algorithms, they found that the algorithm is efficient. The method considerably reduces average execution time and task queue waiting time. Ramezani et al. (2014) proposed a "Task-Based System Load Balancing with Particle Swarm Optimization" approach for achieving a system load balancing by relocating surplus tasks from an overloaded VM instead of migrating the entire overloaded VM. This approach significantly reduces the load-balancing process time compared to traditional load-balancing systems. It reduced VM downtime and the risk of losing a customer's most recent activity while improving cloud customers' Quality of Service. A unique greedy algorithm was given in Paduraru (2014). The researchers concluded that the suggested algorithm outperformed traditional approaches for load balancing algorithms, but it had a higher overhead than other well-known methods. Kapoor and Dabas (2015) suggested a load-balancing technique based on clusters. The algorithm was tested against current and modified throttled algorithms and found superior execution time, response time, throughput, and turnaround time.

The ant colony optimization algorithm was compared to alternative cloud scheduling algorithms FCFS and round-robin. According to the researchers, the ant colony optimization surpassed the FCFS and round-robin algorithms. Awad et al. (2015) announced that their proposed Load Balancing Mutation a Particle Swarm Optimization (LBMPSO) approach was compared to three algorithms: random algorithm, standard PSO, and Longest Cloudlet to Fastest Processor (LCFP). According to them, the comparison shows that LBMPSO

outperformed those algorithms in terms of execution time, makespan, transmission cost, and round trip time. Devi and Uthariaraj (2016) proposed the firefly algorithm, which reduced the time it took to execute the submitted jobs. The proposed approach takes less time to execute than First Come First Served (FCFS). Lu et al. (2016) introduced a hybrid scheduling algorithm DCBT algorithm for load balancing in a distributed environment by merging the methodologies of "Divide-and-Conquer" and "Throttled" algorithms. The researchers defined two situations. The DCBT method was utilized in both situations to schedule incoming client requests to the available virtual machines based on the load on each machine.

The suggested DCBT uses virtual machines better while lowering task execution time. In Mohanty et al. (2017), researchers proposed a metaheuristic load-balancing approach employing Particle Swarm Optimization (MPSO). The proposed method tries to reduce task overhead while maximizing resource utilization. Performance comparisons are done with the genetic algorithm and other algorithms on multiple parameters, including makespan calculation and resource utilization. The proposed method outperforms previous approaches. The imbalance of the load in System Wide Information Management (SWIM) task scheduling is the focus of Li and Wu (2019). A load balancing-based SWIM ant colony task scheduling method (ACTS-LB) is given in that study. The ACTS-LB algorithm outperforms the standard min-min strategy, the ACO algorithm, and the particle swarm optimization (PSO) algorithm in experimental simulations. It not only speeds up job execution and makes better use of system resources, but it also keeps SWIM in a more load-balanced state. According to Kruekaew and Kimpan (2020), virtual machine scheduling management using the "artificial bee colony" (ABC) algorithm and the largest job first (HABC LJF) surpassed ACO, PSO, and IPSO. The results demonstrate that the HABC with the Largest Job First (HABC LJF) heuristic method performs the best in scheduling and load balancing. Sinha and Sinha (2020) developed a load-balancing algorithm for effective resource usage and compared the performance of the proposed algorithms to those of well-known load-balancing algorithms. Compared to the RR, Throttled, ACO, and Hybrid approaches, the EWRR method has less response time.

Singh et al. (2021) employed the Crow intelligent algorithm to distribute tasks among VMs. They evaluated the effectiveness of their method using the CloudSim simulator. 32 GB of Memory and a 1 TB hard drive were employed in 16 virtual machines. They discovered that their system reached an ideal state after a limited number of rounds. Kruekaew and Kimpan (2022) employed the ABC algorithm to improve load balancing and resource utilization. For their performance analysis, they employed CloudSim. Using CloudSim, they compare their results utilizing FCFS, MOPSO (Multi-Objective Particle Swarm Optimization), and MOCS with random data sets. Replication improved the performance of cloud services, according to Javadpour et al. (2023), on one or more virtual machines, they found a solution by choosing and eliminating the VMs whose storage capacity was overcrowded when a VM was requested by more than one user yet needed storage space.

The CloudSim simulator was employed. A greedy algorithm was used for balancing, but the sorting in the algorithm needs time, especially when the job list is very long. So, an enhanced version of Greedy is used in this research.

METHOD

"CloudSim" is a cloud computing simulation program developed in the CLOUDS laboratory at the "University of Melbourne." It allows users to assess individual system concerns without taking into account the low-level details of cloud-based infrastructures and services (Ahmad & Khan, 2019).

CloudSim is an open-source platform for simulating cloud computing services and infrastructure. It enables users to repeatedly test applications under their control, detect system bottlenecks without using real clouds, and experiment with different configurations to build adaptive provisioning ways.

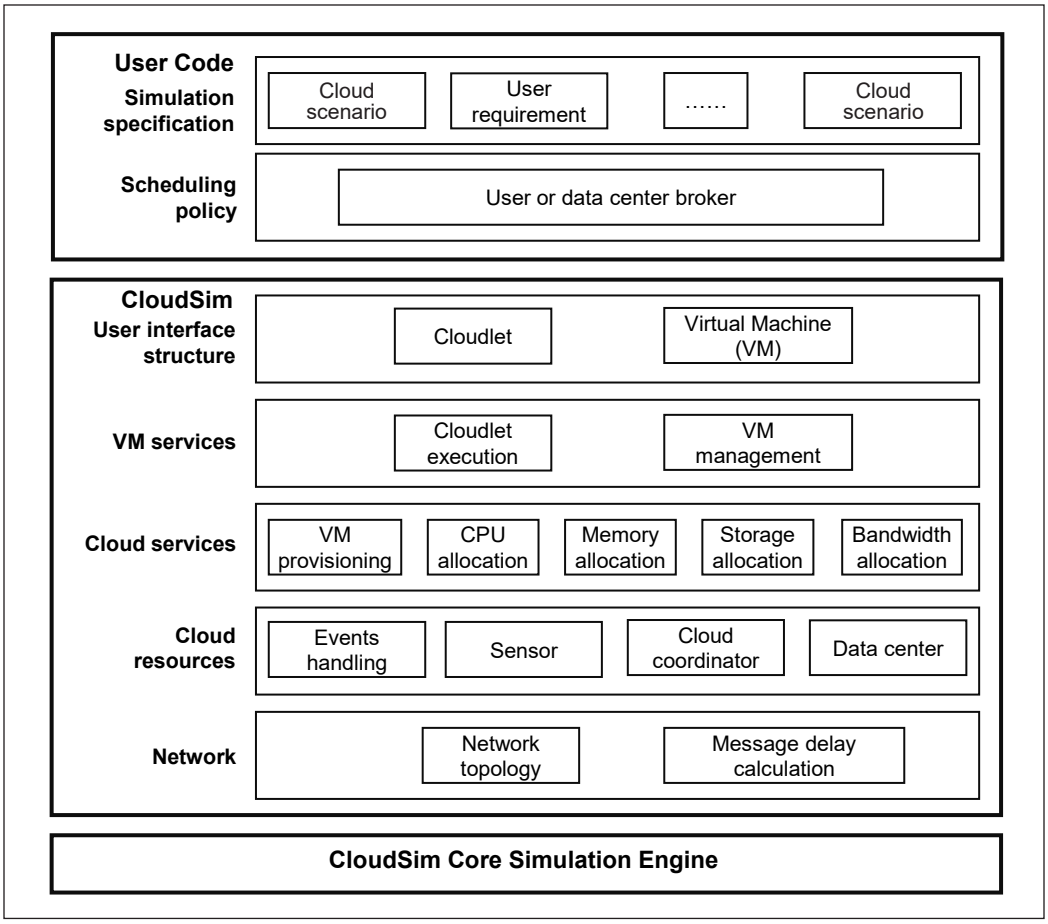

Figure 1. The architecture of CloudSim (Goyal et al., 2012). Adapted from https://doi.org/10.1016/j.proeng.2012.06.412

Version 3.0.1 is utilized in this study. Figure 1 depicts the CloudSim Simulation architecture (Goyal et al., 2012). The simulation layer includes specialized management interfaces for virtual machines, memory, storage, and bandwidth, as well as support for modeling and simulation of virtualized Cloud-based data center infrastructures.

This layer deals with the basics, such as assigning hosts to virtual machines, managing application execution, and keeping track of the system's dynamics (Wang & Wu, 2009). The "User Code" is the top layer of the CloudSim simulation toolkit, and it is the major interface for configuring simulation specifications and characteristics, including the number of machines, apps, jobs, users, and scheduling regulations, as well as their basic structure (Sinha & Shekhar, 2015).

Figure 2 depicts the basic scenario of the CloudSim simulation. Each component of CloudSim is described in depth in this scenario: Data centers (DC) offer resources such as multiple hosts. A host is a hardware machine that can host many virtual machines. A virtual machine (VM) is a piece of logical hardware on which the cloudlet will run. Broker has data center capabilities, which allows it to send virtual machines to the appropriate host. The "Cloud Information Service (CIS)" is in charge of locating resources, indexing them, and calculating the efficiency of data centers (Ahmad & Khan, 2019).

Results can be presented in figures, graphs, tables, and others that make the reader understand easily (Kapoor & Dabas, 2015; Saura et al., 2019). The discussion can be made in several sub-sections.

Two scheduling policies are available in CloudSim: Cloudlet scheduler policies and VMs scheduler policies (Ahmad & Khan, 2019). A greedy algorithm is used in this research to enhance the cloudlet scheduler and obtain a balanced load.

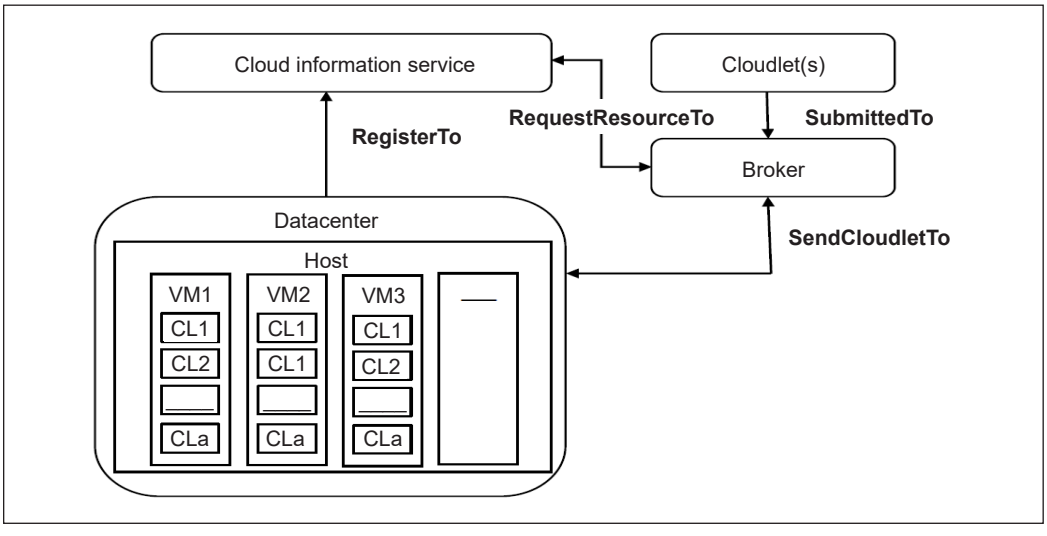

Figure 2. Scenario of CloudSim simulation (Ahmad & Khan, 2019). Adapted from https://doi.org/10.35940/ijrte.B3669.078219

As mentioned before, the current version of the Greedy algorithm needs sorting. It is better to take only the MAX and Min. It will decrease the overhead of the algorithm. The Greedy method proposes allocating the most time-consuming and difficult work to the more efficient virtual machine. i.e., VM with the least load, to rapidly do complicated and tough jobs and reduce the overall system's execution time and makespan.

Makespan refers to the duration of a user's task scheduling. Its value is expressed by the time from the start of the first task to the end of the last task's execution. The performance criteria values were calculated using Equations 1, 2, and 3.

- Makespan for the entire system

$$\text{Makespan} = \text{Max}\ (FT_{ij}) \quad [1]$$

Such as FT_{ij} finishing (ending) time for task T_i. Consider the start time is 0.

- The total capacity of all Virtual machines:

$$C = \sum_{j=1}^{m} C_{vmj} \quad [2]$$

Where C represents the combined capacity of all VMs, the capacity of each virtual machine, or C_{vm}, is determined using Equation 3.

$$C_{vm} = \text{Penum} * \text{Pemips} \quad [3]$$

The number of processing elements (Pe) in VM is called Penum.

The million instructions per second that Pe can execute are known as Pemips.

Following is the *Pseudo code of the greedy algorithm:*

```
Input:  No. of tasks, No. of VMs.
Output: assigning tasks to VMs, system makespan.
Begin
    While the tasks' queue  is not empty Do
    Choose Task T(i) that has the biggest execution time
    Choose VM (j) with Minimum load
    Allocate the task T(i) to Vm(j)
End While
    Calculate the finishing time for all VMs
    Calculate the makespan for all the systems.
End
```

RESULTS AND DISCUSSION

It is contrasted to the popular RR method, which is typically employed for load balancing to improve the performance of the proposed algorithm. The mentioned algorithms in

the related work are not available to us. So, it is impossible to compare with them. The lengths of tasks generated by the random built-in function are presented in Figure 3. The random is chosen because the jobs could arrive at the cloud in any size. The job size is considered in seconds. Cloudsim is used for the simulation of the cloud. The Cloudsim configurations listed below are considered: four VMs are employed for the experiments. The specs are the same for all VMs. As a result, the experiment concentrated on the algorithm rather than the VM requirements. Each virtual machine operates on 250 MIPS, 512 MB of RAM, and 1000 Mbps (Million Byte per second) of bandwidth to run. Indeed, the experiments to test the suggested algorithm use this setting as an example. Ten times, starting at 50 tasks and going up by 50 tasks each time, the experiment is run until it reaches 500 tasks. It will change the system's load and allow researchers to assess how it affects its makespan. Table 1 shows how jobs were completed using the suggested Greedy method. The beginning and ending times of each task, as well as the virtual machine on which it is running. It is independent of task sequencing and instead uses task length to place it on a virtual machine with minimal load.

Table 1
A snapshot of greedy output result for 4 VMs

Cloudlet ID	VM ID	Time of task (sec)	Start time (sec)	Finish time (sec)
6	1	8	0.1	8.1
3	2	12	0.1	12.1
8	1	24	8.1	32.1
0	0	40	0.1	40.1
2	0	8	40.1	48.1
1	3	60	0.1	60.1
5	2	52	12.1	64.1
7	0	36	48.1	84.1
4	3	32	60.1	92.1
11	0	12	84.1	96.1
9	3	4	92.1	96.1

For each number of tasks, the finishing time of every VM and the overall makespan of the system is calculated. The results of both algorithms are shown in Table 2. As seen in the suggested algorithm, every VM

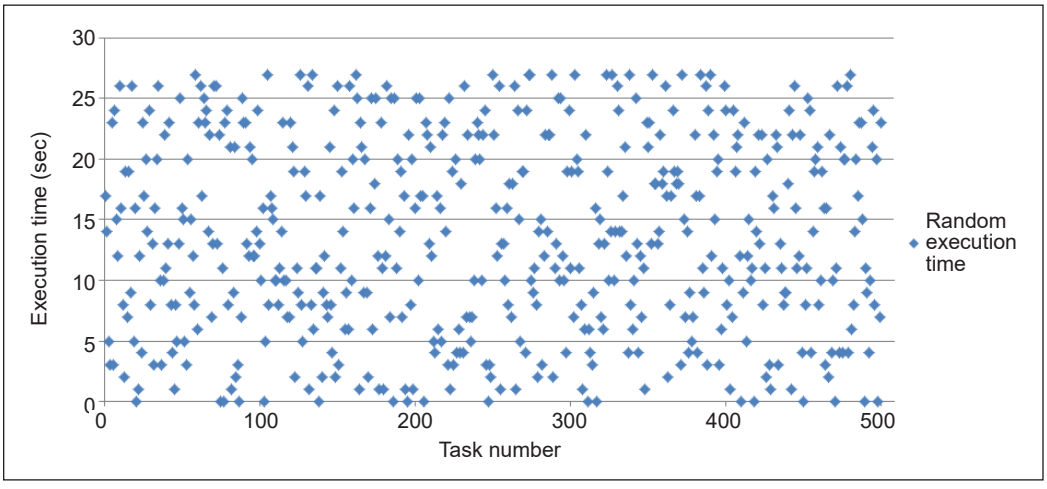

Figure 3. Random execution time

completes its work in a balanced manner (with more or less). However, the difference in the VMs' completion times is evident in RR.

A comparison of the makespan of all systems utilizing RR and the suggested technique is shown in Figure 4. The system's makespan is shorter when using the suggested approach than when utilizing RR. The reason for that is that when the number of tasks increases, there will be a higher probability of the arrival of different sizes of tasks, which in turn improves the selection and gives a better chance to achieve convergent makespan. This behavior differs from traditional RR, which makes no selection but chooses the task from the front of the task queue.

Table 2
VMs finishing times and makespan for both algorithms

	Greedy VMs finishing time (sec)				RR VMs finishing time (sec)				Makespan (sec)	
No. of Task	VM 1	VM 2	VM 3	VM 4	VM 1	VM 2	VM 3	VM 4	Makespan G	Makespan R.R.
50	596.1	596.1	596.1	600.1	652.1	604.1	656.1	476.1	600.1	656.1
100	1408.1	1408.1	1412.1	1412.1	1616.1	1448.1	1204.1	1372.1	1412.1	1616.1
150	2156.1	2156.1	2156.1	2156.1	2120.1	2160.1	1948.1	2396.1	2156.1	2396.1
200	2920.1	2920.1	2920.1	2920.1	2940.1	2864.1	3156.1	2720.1	2920.1	3156.1
250	3672.1	3672.1	3672.1	3672.1	3604.1	3568.1	3856.1	3660.1	3672.1	3856.1
300	4072.1	4076.1	4076.1	4076.1	3964.1	4188.1	4040.1	4108.1	4076.1	4188.1
350	5172.1	5172.1	5172.1	5172.1	5228.1	5260.1	5276.1	4924.1	5172.1	5276.1
400	5824.1	5824.1	5828.1	5828.1	6276.1	5548.1	5720.1	5760.1	5828.1	6276.1
450	6324.1	6328.1	6328.1	6328.1	6208.1	6344.1	6432.1	6324.1	6328.1	6432.1
500	7324.1	7324.1	7328.1	7328.1	6912.1	7888.1	7344.1	7160.1	7328.1	7888.1

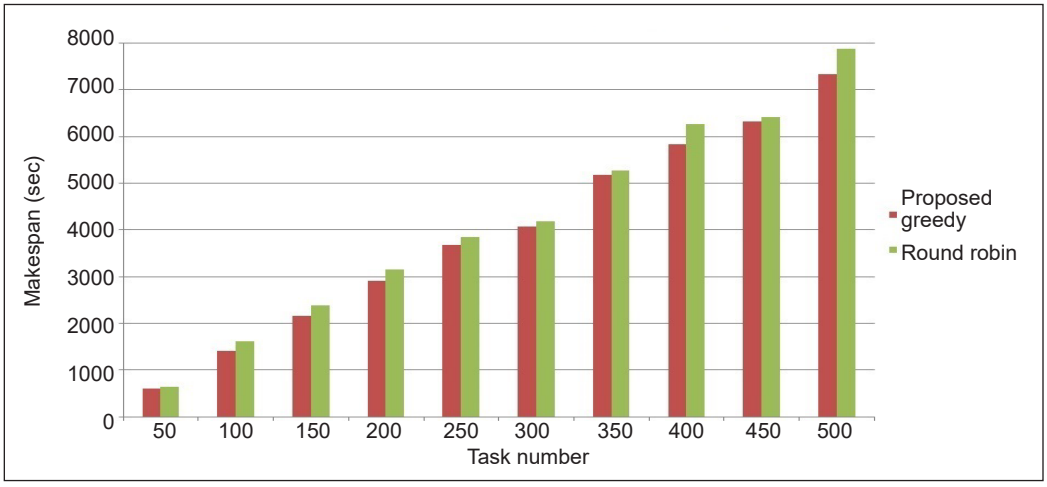

Figure 4. Makespan for different no. of tasks for round robin and proposed greedy algorithm

Figure 5 shows that all VMs in RR finish their jobs at various times. Whereas in greedy, VMs finish jobs nearly simultaneously (Figure 6). The greedy allocator takes the initial step in load balancing by assigning the task to the VM with the lowest load.

According to Table 3 and Figure 7, the improved greedy algorithm's standard deviation of the finishing values spans from 0 to 2.309401, whereas the RR algorithm's standard deviation ranges from 84.0555372 to 414.3750314. They also demonstrate no correlation between the number of tasks and the standard deviation of the VMs' completion values.

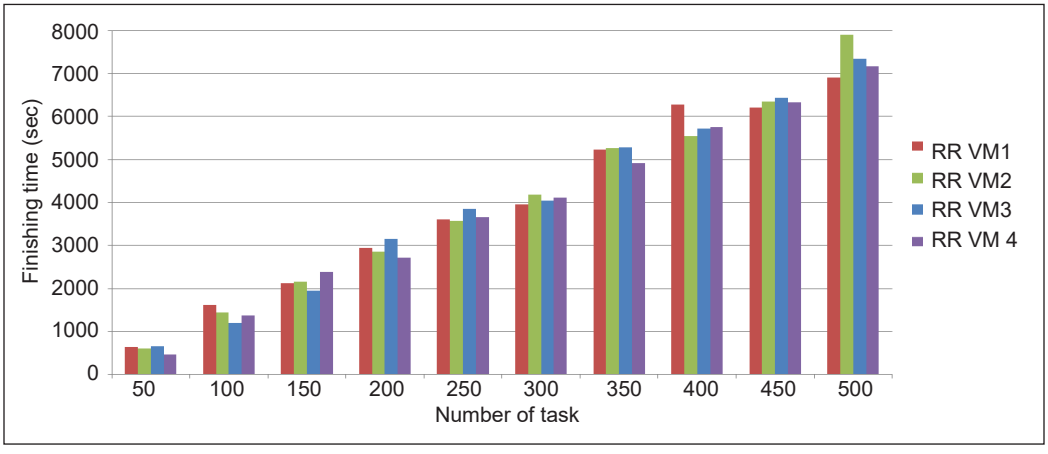

Figure 5. Unbalanced Virtual Machines' (VM) finishing time in round robin (RR)

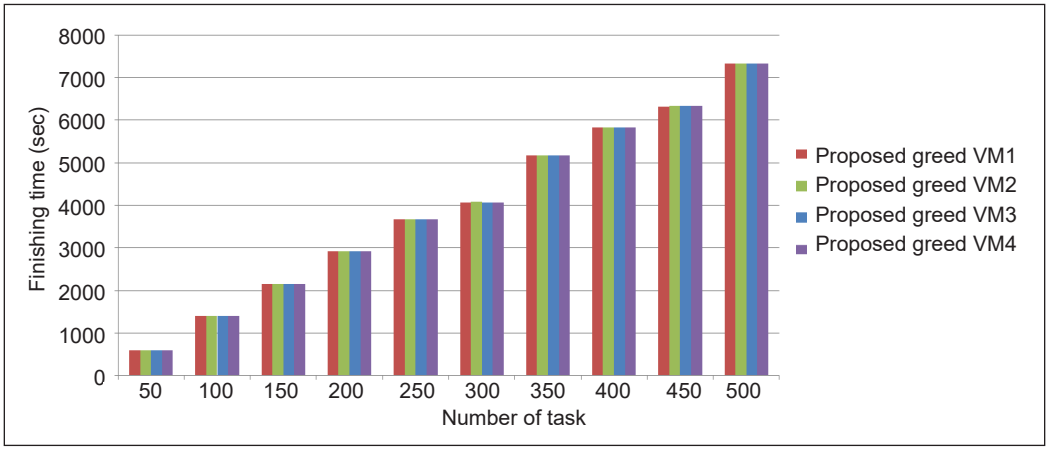

Figure 6. Balanced Virtual Machines' (VM) finishing time in greedy algorithm

Table 3
Enhanced greedy and round robin algorithm standard deviation

No. of tasks	50	100	150	200	250	300	350	400	450	500
STDEV Greedy	2.00	2.31	0.00	0.00	0.00	2.00	0.00	2.31	2.00	2.31
STDEV RR	84.06	171.04	184.55	181.87	128.37	95.58	166.53	313.78	92.17	414.38

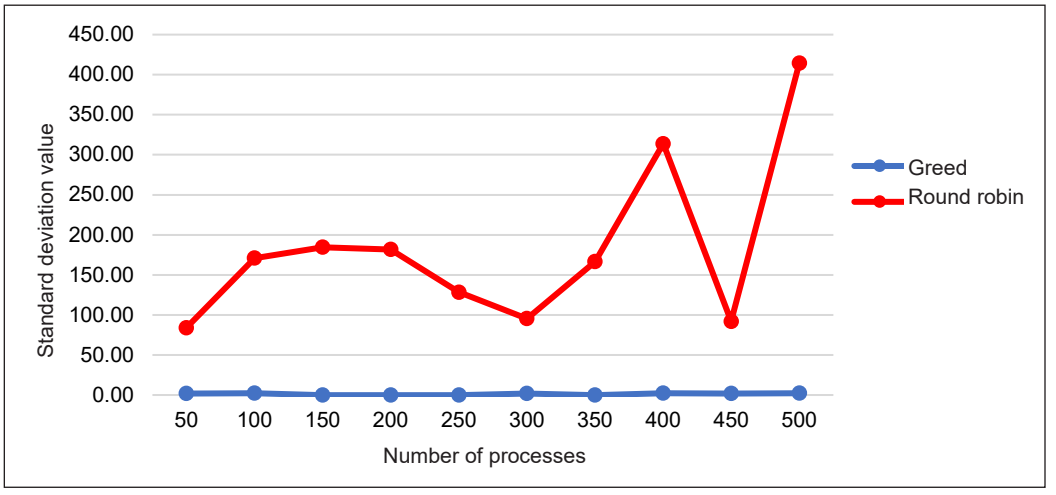

Figure 7. Standard deviation of makespans at different loads

CONCLUSION

The present research proposes an enhanced version of a Greedy algorithm to allocate tasks in the cloud. When the request arrives at the data center, deciding which VM is the most suitable to execute it is necessary. A greedy algorithm will help to make this choice to achieve load balancing. There is no need for migration since the chosen VM will satisfy its constraints. The algorithm was compared to RR, and the results showed that the makespan of the system using Greedy is less than its equivalent using RR. Also, all VMs in Greedy finish their work simultaneously, whereas RR VMs vary in finishing time, leading to an unbalanced system load.

ACKNOWLEDGMENTS

The authors appreciate the cooperation of the University of Mosul, Iraq, in completing the experimental work.

REFERENCES

Ahmad, M. O., & Khan, R. Z. (2019). Cloud computing modeling and simulation using cloudsim environment. *International Journal of Recent Technology and Engineering*, *8*(2), 5439-5445. https://doi.org/10.35940/ijrte.B3669.078219

Awad, A. I., El-Hefnawy, N. A., & Abdel-Kader, H. M. (2015). Enhanced particle swarm optimization for task scheduling in cloud computing environments. *Procedia Computer Science*, *65*, 920-929. https://doi.org/10.1016/j.procs.2015.09.064

Babu, L. D. D., & Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing Journal*, *13*(5), 2292-2303. https://doi.org/10.1016/j.asoc.2013.01.025

Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., & Moscardelli, L. (2011). Tight bounds for selfish and greedy load balancing. *Algorithmica, 61*, 606-637. https://doi.org/10.1007/s00453-010-9427-8

Coady, Y., Hohlfeld, O., Kempf, J., McGeer, R., & Schmid, S. (2015). Distributed cloud computing: Applications, status quo, and challenges. *Computer Communication Review*, *45*(2), 38-43. https://doi.org/10.1145/2766330.2766337

Dave, S., & Maheta, P. (2014). Utilizing round robin concept for load balancing algorithm at virtual machine level in cloud environment. *International Journal of Computer Applications*, *94*(4), 23-29. https://doi.org/10.5120/16332-5612

Devi, D. C., & Uthariaraj, V. R. (2016). Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. *Scientific World Journal*, *2016*, Article 3896065. https://doi.org/10.1155/2016/3896065

Fatima, S. G., Fatima, S. K., Sattar, S. A., Khan, N. A., & Adil, S. (2019). Cloud computing and load balancing. *International Journal of Advanced Research in Engineering and Technology*, *10*(2), 189-209. https://doi.org/10.34218/IJARET.10.2.2019.019

Goyal, T., Singh, A., & Agrawa, A. (2012). Cloudsim: Simulator for cloud computing infrastructure and modeling. *Procedia Engineering*, *38*, 3566-3572. https://doi.org/10.1016/j.proeng.2012.06.412

Javadpour, A., Sangaiah, A. K., Pinto, P., Ja'fari, F., Zhang, W., Abadi, A. M. H., & Ahmadi, H. R. (2023). An energy-optimized embedded load balancing using DVFS computing in cloud data centers. *Computer Communications*, *197*, 255-266. https://doi.org/10.1016/j.comcom.2022.10.019

Kapoor, S., & Dabas, C. (2015). Cluster based load balancing in cloud computing. In *2015 8th International Conference on Contemporary Computing, IC3 2015* (pp. 76-81). IEEE Publishing. https://doi.org/10.1109/IC3.2015.7346656

Kruekaew, B., & Kimpan, W. (2020). Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. *International Journal of Computational Intelligence Systems*, *13*(1), 496-510. https://doi.org/10.2991/ijcis.d.200410.002

Kruekaew, B., & Kimpan, W. (2022). Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. *IEEE Access*, *10*, 17803-17818. https://doi.org/10.1109/ACCESS.2022.3149955

Kumar, K. P., Ragunathan, T., Vasumathi, D., & Prasad, P. K. (2020). An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. *International Journal of Intelligent Engineering and Systems*, *13*(3), 422-432. https://doi.org/10.22266/IJIES2020.0630.38

Kumar, P., & Kumar, R. (2019). Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Computing Surveys*, *51*(6), Article 120. https://doi.org/10.1145/3281010

Li, G., & Wu, Z. (2019). Ant colony optimization task scheduling algorithm for SWIM based on load balancing. *Future Internet*, *11*(4), Article 90. https://doi.org/10.3390/fi11040090

Lu, Y., Zhang, J., Wu, S., Zhang, S., Zhang, Y., Li, Y., Ghosh, S., Banerjee, C., Kulkarni, A. K., Annappa, B., Domanal, S. G., Reddy, G. R. M., Komarasamy, D., & Muthuswamy, V. (2016). Load balancing in cloud environment using a novel hybrid scheduling algorithm. In *2015 IEEE International Conference on Cloud*

Computing in Emerging Markets, CCEM 2015 (pp. 37-42). IEEE Publishing. https://doi.org/10.1109/CCEM.2015.31

Mishra, S. K., Sahoo, B., & Parida, P. P. (2018). Load balancing in cloud computing: A big picture. *Journal of King Saud University - Computer and Information Sciences, 32*(2), 149-158. https://doi.org/10.1016/j.jksuci.2018.01.003

Mohanty, S., Patra, P. K., Ray, M., & Mohapatra, S. (2017). A novel meta-heuristic approach for load balancing in cloud computing. *International Journal of Knowledge-Based Organizations*, *8*(1), 29-49. https://doi.org/10.4018/ijkbo.2018010103

Nerkar, M. H. (2012). Cloud computing in distributed system. *International Journal of Computer Science and Informatics*, *1*(10), 97-101. https://doi.org/10.47893/ijcsi.2012.1072

Paduraru, C. I. (2014). A greedy algorithm for load balancing jobs with deadlines in a distributed network. *International Journal of Advanced Computer Science and Applications*, *5*(2), 56-59. https://doi.org/10.14569/ijacsa.2014.050209

Ramezani, F., Lu, J., & Hussain, F. K. (2014). Task-based system load balancing in cloud computing using particle swarm optimization. *International Journal of Parallel Programming*, *42*(5), 739-754. https://doi.org/10.1007/s10766-013-0275-4

Saura, J. R., Herraez, B. R., & Reyes-Menendez, A. (2019). Comparing a traditional approach for financial brand communication analysis with a big data analytics technique. *IEEE Access*, *7*, 37100-37108. https://doi.org/10.1109/ACCESS.2019.2905301

Singh, H., Tyagi, S., & Kumar, P. (2021). Cloud resource mapping through crow search inspired metaheuristic load balancing technique. *Computers and Electrical Engineering*, *93*, Article 107221. https://doi.org/10.1016/j.compeleceng.2021.107221

Sinha, G., & Sinha, D. (2020). Enhanced weighted round robin algorithm to balance the load for effective utilization of resource in cloud environment. *EAI Endorsed Transactions on Cloud Systems*, *6*(18), Article 166284. https://doi.org/10.4108/eai.7-9-2020.166284

Sinha, U., & Shekhar, M. (2015). Comparison of various cloud simulation tools available in cloud computing. *International Journal of Advanced Research in Computer and Communication Engineering*, *4*(3), 171-176. https://doi.org/10.17148/ijarcce.2015.4342

Tawfeek, M. A., El-Sisi, A., Keshk, A. E., & Torkey, F. A. (2013). Cloud task scheduling based on ant colony optimization. In *2013 8th International Conference on Computer Engineering & Systems (ICCES)* (pp. 64-69). IEEE Publishing. https://doi.org/10.1109/ICCES.2013.6707172

Wang, Y. H., & Wu, I. C. (2009). Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms. *Software - Practice and Experience*, *39*(7), 701-736. https://doi.org/10.1002/spe