

Cloud service analysis using round-robin algorithm for quality-of-service aware task placement for internet of things services

Nor Syazwani Mohd Pakhrudin¹, Murizah Kassim^{1,2}, Azlina Idris¹

¹School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, Shah Alam, Malaysia

²Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Universiti Teknologi MARA, Shah Alam, Malaysia

Article Info

Article history:

Received Jul 4, 2022

Revised Sep 8, 2022

Accepted Oct 13, 2022

Keywords:

CloudSim

Central processing unit
scheduling

Internet of things services

Load-balancing

Quality of service aware

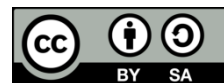
Round-robin

Virtual machines

ABSTRACT

Round-robin (RR) is a process approach to sharing resources that requires each user to get a turn using them in an agreed order in cloud computing. It is suited for time-sharing systems since it automatically reduces the problem of priority inversion, which are low-priority tasks delayed. The time quantum is limited, and only a one-time quantum process is allowed in round-robin scheduling. The objective of this research is to improve the functionality of the current RR method for scheduling actions in the cloud by lowering the average waiting, turnaround, and response time. CloudAnalyst tool was used to enhance the RR technique by changing the parameter value in optimizing the high accuracy and low cost. The result presents the achieved overall min and max response times are 36.69 and 650.30 ms for running 300 min RR. The cost for the virtual machines (VMs) is identified from \$0.5 to \$3. The longer the time used, the higher the cost of the data transfer. This research is significant in improving communication and the quality of relationships within groups.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Murizah Kassim

Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Universiti Teknologi MARA

40450 Shah Alam, Selangor, Malaysia

Email: murizah@uitm.edu.my

1. INTRODUCTION

Task scheduling is one of the most important things to learn in a cloud computing environment. Due to the rising number of cloud users, service providers must priorities generating profitability while ensuring sufficient access to remote resources. The improved technique is used in task scheduling to match client tasks with relevant and accessible virtualized resources. Task scheduling algorithms are a collection of rules and regulations that are used to assign jobs to the appropriate resources like central processing unit (CPU), memory, and bandwidth to maximize performance and resource usage [1]. Scheduling is a fundamental feature in operating systems because almost all computer resources are scheduled before they are used [2].

It requires devoting resources and effort to specific processes or tasks to meet performance goals. Different processes on a computer with multi-programming compete for the CPU simultaneously. The round-robin (RR) algorithm was developed primarily for time-sharing systems and is one of the most popular scheduling algorithms. It is one of the oldest, easiest, and most reasonable algorithms available. The processes in this algorithm share the CPU time by allocating a quantum time (QT) slice to each step in QT. It normally lasts between 10 and 100 milliseconds [3]. The tasks will be distributed among the virtual machines (VMs) in the RR method. The RR algorithm has been used in a variety of settings. It is used in CPU scheduling, as previously stated, to allocate CPU time among the processes. It is used in the cloud computing environment to boost system efficiency and user quality of service (QoS). It is the time interval between the start of a process

and its accomplishment [4]. The RR algorithm's efficiency is entirely dependent on quantum mechanics. Because of its simplicity and fairness, the RR algorithm is one of the most used scheduling algorithms. VM is an essential processing unit based on cloud computing. The unit in charge of controlling all VMs is known as the VM manager (VMM) [5]. Though, cloud storage has numerous issues, such as resource scheduling, load balancing, security, privacy, and QoS management. The RR algorithm's performance depends on the time quantum used. There have been many efforts to pick the best time quantum [6]. Due to low performance, real-time operating systems apart from RR scheduling are infrequently employed in these algorithms [7]. Optimization of the RR scheduling method revealed its efficacy [8].

In cloud computing, the cloud infrastructure cannot handle the flow of information independently with the profusion of data, devices, and interactions [9]. Load balancing distributes all workloads across each node in a shared or mutual system to maximize resource utilization and reduce job response time. The most important aspect of cloud computing is scheduling, which includes workloads and workflow scheduling under the platform as a service model. The infrastructure as service task to VMs scheduling which machine is decided by the scheduler should go on which job or VMs [10]. Evaluating the energy required for communication between the devices participating in this process and the suggested method's appropriateness for handling optimization problems like VM placement is necessary [11]. Load balancing is a method of reassigning the entire load to the various nodes of a collaborative system to improve resource efficiency and the job's response time while avoiding a situation where specific nodes are overloaded and others are underloaded. A load-balancing algorithm involved in identity ignores the system's previous state or behavior, relying instead on the system's neighboring behavior. This load can be measured in terms of CPU, memory use, sluggishness, or network load [12]. Load balancing's primary aim is to enhance device efficiency and functionality for today's QoS for the internet of things (IoT) services in communication and data transfer. It is used in cloud computing systems to provide successful alternative solutions in the event of a system failure and to ensure the best possible use of system components. Load balancing techniques are divided into two main models: static model and dynamic model. First is static load balancing: static load balancing occurs in a static environment where the output of algorithms is unaffected by the system's current state. As a result, user expectations do not change over time [13]. Second is dynamic load balancing: the system's state has a significant impact on balancing the efficiency of the algorithms. Since resources are versatile in a dynamic environment, algorithms efficiently perform load balancing [14].

Service broker techniques reduce user request latency and delegate user requests to the required data center. Only the VMs inside the data center of the cloud system communicate with the data center controller. Closest data center dynamically reconfigured routing, and performance-optimized routing is several well-known strategies used in cloud environments [15]. The closest data center is one of the most straightforward techniques for simulating the cloud environment; it uses the generated index to store information about the available virtualized data centers and operates the nearest data center based on its latency for request execution [16]. The service broker uses the performance-optimized routing technique to calculate the performance of all the necessary attributes in the data centers. The request is then sent to the designated data center to achieve the best latency possible [17]. Configuring dynamic routing: the service broker oversees the cloud application's scalability. This strategy is an established scheme for service broker policy (SBP) that controls the number of VMs operating on every virtualized data center. The ability to monitor the number of VMs from the data center would provide the best system efficiency in processing time for requests, especially to maintain the QoS for IoT services [18].

Cloud computing is Internet-based computing that offers a pool of customizable computing resources such as networks, storage, servers, applications, and services without requiring interaction with the service provider and with little administration effort [19]. The RR protocol's moving horizon estimation problem for a class of discrete time-delay systems. To avoid data collisions, communication between the sensor nodes and the remote state estimator is carried out over a shared network, with only one sensor node able to transmit data at any given time. The RR protocol organizes the transmission order of sensor nodes, with the selected node gaining network access modeled as a periodic function. The device model is reformulated into a linear system without delays due to lifting technology. The problem at hand aims to construct a moving horizon estimator that minimizes the estimation error. In matrix inequality, a proper condition is defined to ensure ultimate boundedness [16].

Two optimization problems are proposed within the existing theoretical framework to measure the corresponding estimator parameters according to two different performance requirements, such as the smallest ultimate bound and the fastest decay rate. Finally, simulation examples are given to demonstrate the utility of the estimator design scheme [20]. In interactive systems, an adjustable RR scheduling algorithm was proposed. Within the operating system, CPU scheduling is regarded as a fundamental task. A comparison of CPU scheduling algorithms has been suggested using scheduling parameters such as waiting time, context switches, and others. This paper presents a modified version of the RR algorithm as an attempt to combine the benefits of RR's low scheduling overhead with a preference for short processes to reduce average waiting time and the number of context changes in interactive (time-shared) systems. When the time slice defined by the RR policy

expires, a threshold is used to decide if the running phase will be interrupted or will continue to run until it is terminated. Compared to the RR algorithm, the suggested improvement reduces the average waiting time and the number of context switches [21].

The RR scheduling algorithm can perform better by selecting a suitable time quantum. The RR algorithm described in this study has a proficient time quantum that was calculated by taking into account the greatest possible difference between the differences of neighboring consecutive processes in the ready queue. The suggested method aims to improve system performance and RR results. In terms of testing and comparison, the algorithm performs better than the RR techniques mentioned in this work. The comparison to the mentioned techniques, it decreases the average turn-around time, average waiting time, number of context switches, and other CPU scheduling requirements [22]. The algorithm policy employed by a CPU to schedule running tasks affects the performance of an operating system (OS). This study compares four RR algorithms: Adaptive RR algorithm, best time quantum RR CPU scheduling, optimal RR scheduling using the Manhattan distance algorithm, and improved RR scheduling algorithm. Four performance indicators were used to compare these algorithms: the amount of context switching (NCS), average waiting time (AWT), average turnaround time (ATT), and average response time (ART) [23]. The simulation results show that both the adaptive RR and optimal RR scheduling using Manhattan distance algorithms are more effective to use because they report the lowest performance factor values [23].

A predictive modified RR scheduling algorithm for web server clusters. As dynamic content transforms conventional web environments, the need for high-performance web servers grows, resulting in the usage of cluster-based web servers. As a result, effective and equal load balancing of cluster web servers is a critical challenge, particularly with the emergence of dynamic content and database-driven applications on the internet, such as e-commerce and corporate databases. They propose a predictive modified RR load-balancing algorithm that applies the mythology of prediction to the load-balancing sector after stating the problem discussed in this paper and some preliminaries. They use simulation results to check the proposed load-balancing algorithm's effectiveness. In comparison to RR and modified RR scheduling, the algorithm dramatically reduces both the load range and load variance [24]. Cloud infrastructure, a modern computer system, allows users to get pooled resources when they need them to use them when they want. The distributing workload among computing system nodes is known as load balancing. This paper provided updated RR and modified Honeybee algorithms for load balancing based on honeybee and RR foraging behaviors (VM) to regulate load through virtual computers. Honeybees are jobs that have been removed from congested VMs. Tasks in VM queues are examined to determine their aims to achieve a fast response time and a minimal number of task migrations using the suggested protocol. The test results reveal a considerable improvement in QoS [25].

This work aimed to provide some recommended techniques and scheduling algorithms for resource allocation in cloud computing via data center virtualization. The article also intends to investigate the function of virtualization in successfully delivering resources based on customers' needs. Results indicated that these approaches demonstrate that each proposed technique and scheduling algorithm has a straightforward process for maximizing cloud data center resources. Using virtualization as a strategy, the researchers found that they could improve network speed while saving money by lowering the number of physical machines (PMs) in the data center. They also found that they could balance workloads, save energy, and actively distribute resources to meet the needs of their clients. According to our findings, the availability of VM resources and the time it takes to execute requests are the most important aspects to consider in any effective resource allocation method. As a consequence of our analysis of the recommended methodologies, we've discovered that request execution time and VM availability are critical factors that should be considered in any resource allocation strategy [26].

RR is a computer networking protocol that allows multiple hosts to communicate with each other. Related work includes the development of protocols such as transmission control protocol (TCP) and user datagram protocol (UDP). These protocols allow computers to communicate with each other and make it possible for websites to be accessed by different devices. RR is still used in many networks and is considered a standard in RR scheduling, the time quantum is limited, and no process gets more than a one-time quantum at once. If the time quantum is too big, the process reaction time may be too long for interactive environments. If the time quantum is too short, context switches occur too often, increasing overhead and reducing performance. In this paper, the original RR method was the modifications based on the basic RR regarding average waiting time and turnaround time. The suggested model improves the functionality of the current RR method for scheduling activities in the cloud by lowering the average waiting time, turnaround time, and response time.

2. THE PROPOSED WORK

The research technique utilized to gather and evaluate the data needed for this study is described in further detail in this section. A description of the study design comes first in this section, and a simulation of

the data collection process comes next. The task of providing QoS guarantees to users is difficult and causes errors process in the current cloud systems. To address this, we propose a method of cloud service analysis that uses an RR algorithm to automatically place tasks in an optimal manner that is aware of QoS requirements. This algorithm has the potential to improve the QoS for users by automatically considering their service-level agreements (SLAs) when placing tasks. In addition, the proposed method is scalable and can be used to analyze large cloud systems. In this paper, we use the RR algorithm to place tasks in an optimal manner so that they meet user demands for QoS. The algorithm first identifies a task optimal with the way the that highest takes priority such as QoS requirements. We show that the RR algorithm can improve the QoS for users by ensuring that tasks are placed in an order that best meets their QoS requirements.

Figure 1 shows the suggested CPU scheduling calculation is based on a RR scheduling calculation modification. The proposed algorithm's stages are to maintain the procedures in the ready queue as they appear. Then, compute the CPU burst time for many processes. If the VM status is available and the process has the same burst, proceed. The next step is to assign the CPU for the duration of the first task in the ready queue. Expel the current process from the organized queue and place it at the end of the queue for future execution if the remaining burst duration is greater than the time quantum. Then select the following process in the ready queue and assign the CPU to it for the duration of the procedure. Repeat until the process queue is empty. Calculate the average waiting time and average turnaround time for each procedure.

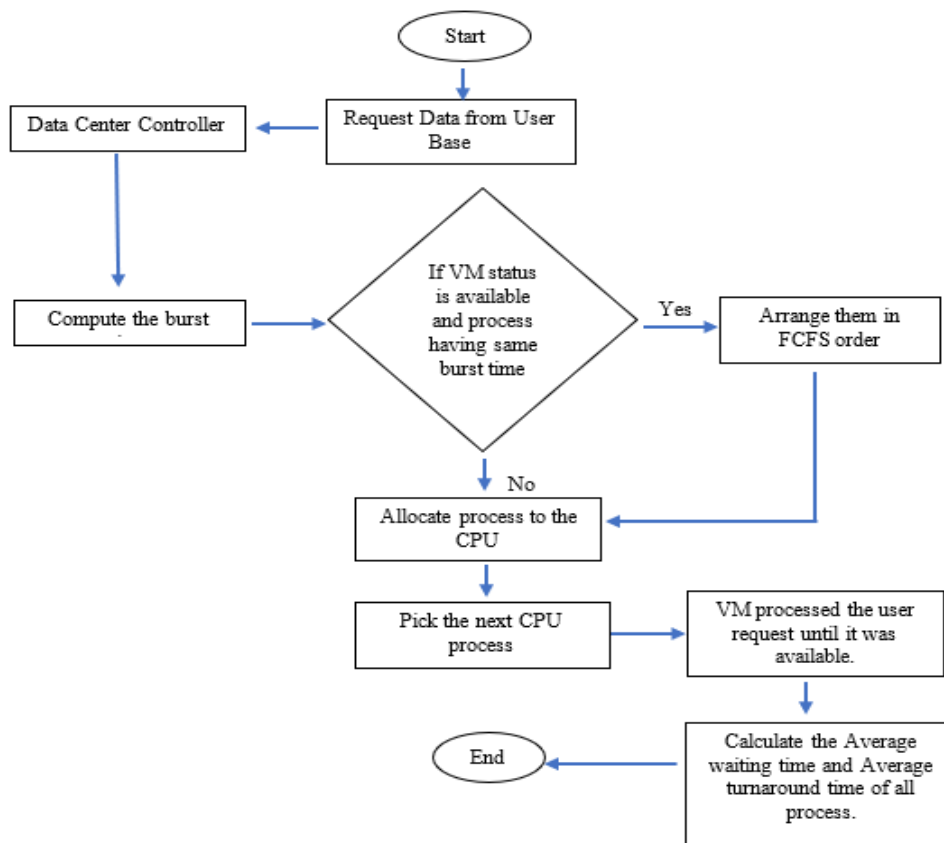


Figure 1. Flowchart for the proposed algorithm

3. METHOD

The methodology section presents the mathematical model to conduct the RR scheduling algorithm. Then, the simulation using CloudAnalyst is presented. Lastly, the parameter of the RR CloudAnalyst is derived, and the simulation map implemented in Malaysia from the CloudAnalyst is presented.

3.1. Mathematical formula round-robin scheduling algorithm

The RR scheduling algorithm is a popular scheduling algorithm used by many operating systems. The algorithm works by assigning each process a time slice, or quantum, in which it is allowed to run. Once a process has used up its time slice, it is prevented and moved to the end of the queue. The process then waits its turn to run again. The key benefit of the RR algorithm is to ensure that each process has a fair chance to run.

This is important because it prevents processes from becoming overloaded and stressed out, which can lead to problems such as crashes or hardware failures. Additionally, the RR algorithm is relatively quick and easy to implement, so it can be used in a wide variety of situations. Let P_1, P_2, \dots, P_n , be n-processes ordered in a ready queue (to be referred to as RQ) according to the first come-first serve algorithm, with $BT_i; i = 1, 2, 3, \dots, n$, as their predicted CPU bursts for completion [27]. The average turnaround time for a method P_i can be determined by adding up the total time spent in RQ.

$$\begin{aligned} \text{Turnaround Time of } P_1 &= BT_1 \\ \text{Turnaround Time of } P_2 &= BT_1 + BT_2 \end{aligned}$$

Also,

$$\text{Turnaround Time of } P_i = BT_1 + BT_2 + \dots + BT_i$$

Hence, (1) is obtained for P_i (TAT)/

$$P_i(\text{TAT}) = \sum_{k=1}^i BT_k - AT_i \quad (1)$$

Average turnaround time of a process P_i can be calculated with the help of (2).

$$\text{Average Turnaround Time} = \frac{\sum_{k=1}^n P_k(\text{TAT})}{n} \quad (2)$$

For total waiting time for a process, P_i (P_i (TWT)) can be calculated by summing up the total time devoted to the process P_i in the RQ.

$$\begin{aligned} \text{Waiting Time of } P_1 &= 0 \\ \text{Waiting Time of } P_2 &= BT_1 \\ \text{Waiting Time of } P_3 &= BT_1 + BT_2 \end{aligned}$$

Hence, in general for the i process, we have

$$\text{Waiting Time of } P_i = BT_1 - 1 + BT_2 - 2 + \dots + BT_i$$

Hence, (3) is obtained for P_i (TWT).

$$P_i(\text{TWT}) = \sum_{k=1}^{i-1} BT_k - AT_i \quad (3)$$

Average waiting time of a process P_i can be calculated with the help of (4).

$$\text{Average Waiting Time} = \frac{\sum_{k=1}^n P_k(\text{TWT})}{n} \quad (4)$$

For the priority scheduling algorithm, P_1, P_2, \dots, P_n , are n-processes arranged in first come-first serve in a temporary queue (referred to as TQ), with BT_i and $Pr_i; i=1, 2, 3, \dots, n$, as their predicted CPU burst time and process priority, respectively. With the aid of the ceiling function [], we were able to estimate the total number of context switches. The ceiling function maps a real number to the smallest integer that comes after it. Quantum time is still unchanged in the RR method, and processes join the queue with their CPU time requirements. The total number of cycles needed to complete a process is calculated by dividing the process's burst time by its quantum time [28].

3.2. Simulation in the CloudAnalyst

CloudAnalyst is a cloud simulator program that developers use to simulate the best operational methods to improve the cloud service platform quality and decrease the total ownership cost (TOC). It also aids cloud service providers in the creation and development of their own virtualized data centers all over the world. Our proposed work introduces the CloudAnalyst Tool, which adds flexible parameter values and additional choices from the CloudSim toolbox for doing high-precision, low-cost simulations. The CloudAnalyst platform consists of numerous interdependent parts, including a data center controller, region presenter, cloud service broker, graphical user interface (GUI) platform, internet simulator, user database, and virtual machine load balancer.

One of the simplest uses of a service broker is using a routing strategy known as service proximity-based routing. The proximity service broker maintains an index database of all data centers divided by geographic region. The internet looks up the destination data center controller in the service proximity service broker when it receives a request from a user. After obtaining the sender's location, the proximity service broker searches the internet characteristics for the proximity list for that area. This list sorts the other regions from lowest to highest network latency when computing network latency from the selected region. The proximity service broker chooses the first data center in the proximity list and the first data center in the proximity list located in the earliest or most distant location. One is randomly chosen if an area has more than one data center. The internet characteristics screen can set internet latency and bandwidth parameters. For these two groups, it provides two matrices. The output of the RR algorithm was examined using the CloudAnalyst simulation tool. The framework of CloudAnalyst has many features, including the ability to configure the position of the data center depending on your needs, the location of the user who develops the applications, and the ability to run several simulations by adjusting small parameters. We divided the world into five regions for the simulation testing, with different modeling for each area as shown in Table 1, suggesting that the number of users varies by location, suggesting that the number of users varies by location. The limitation of the simulation's software is that a world map is used to define the cloud nodes in measuring the distances between the 5-resource server connected to one main server. Thus, the country is like 0 – S. West Malaysia, 1 – S. East Malaysia, 2 – E. West Malaysia, 3 – N West Malaysia, 4 – Center of West Malaysia, and 5 – N. East Malaysia are used on the map.

Table 1. Parameter of the RR CloudAnalyst

User base (UB)	Region	Requests per User per Hour	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Peak Users
UB1	0 – S. West Malaysia	60	100	3	9	1000	100
UB2	1 – S. East Malaysia	60	100	3	9	1000	100
UB3	2 – E. West Malaysia	60	100	3	9	1000	100
UB4	3 – N West Malaysia	60	100	3	9	1000	100
UB5	4 – Center of West Malaysia	60	100	3	9	1000	100
UB6	5 – N. East Malaysia	60	100	3	9	1000	100

After all the parameters have been set up to construct a simulation configuration, the user must return to the main screen and select “Run Simulation” from the control panel to run the simulation. The simulation will begin, and the percentage completion will be shown in the progress bar at the top of the simulation screen. An introductory animation will be shown on the simulation screen to illustrate which user bases are sending messages to which data centers as shown in Figure 2.

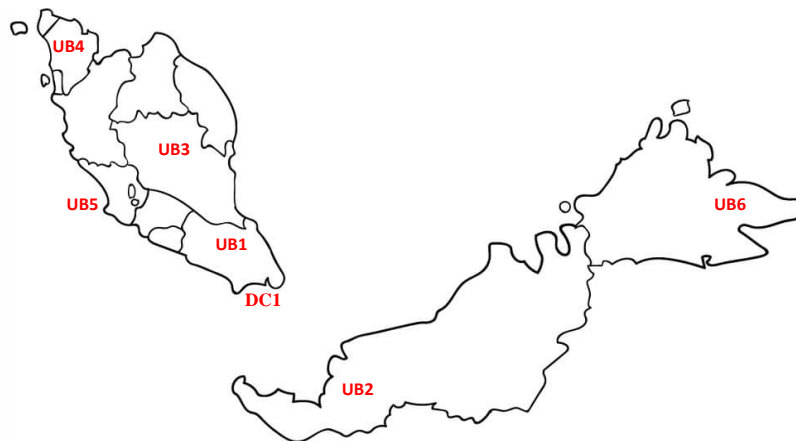


Figure 2. The simulation map from the CloudAnalyst

4. RESULT AND DISCUSSION

Figure 3 represents the overall response time summary results between 60 to 300 min. The results show the overall for the 6 user bases using the RR algorithm. The graph in Figure 3 shows that the highest average for the overall response time summary is 293.82 ms at 300 min. The min and max overall response times are 36.69 and 650.30 ms for 300 min.

Figure 4 shows the processing time between the data center processing time and data center request servicing times (DC1). The results show that both the data center processing time and data center request servicing times (DC1) have the same results for the speeds for processing time using 6 Userbase. The average processing time for both is 0.28 to 0.33 ms. The highest average processing time at the speed of 0.33 at 300 min.

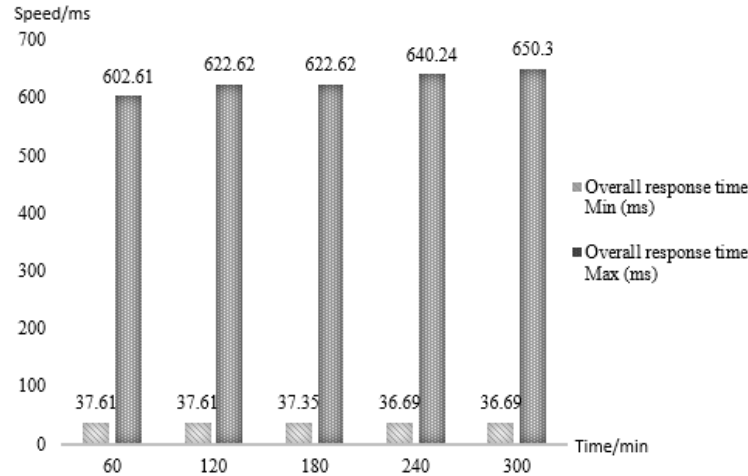


Figure 3. Analysis of response time for the RR

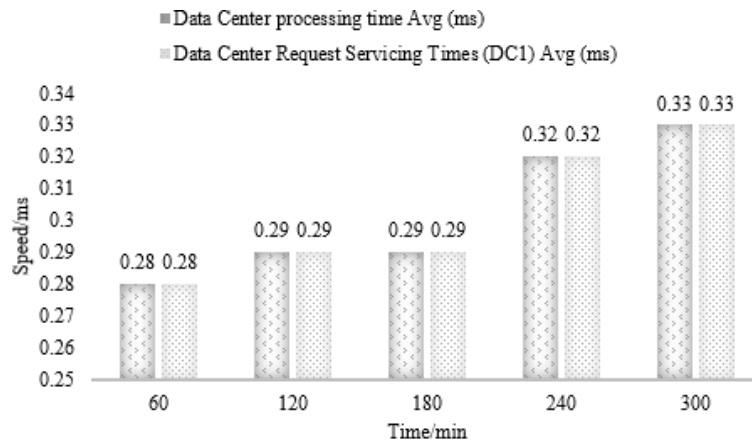


Figure 4. Analysis of the processing time

Considering the cost-effectiveness of the algorithm is an important part of any performance study, and therefore cost-benefit analysis is taken into account. Since the overall cost of each algorithm is the same, we can determine the total cost of data transfer as follows.

$$Total\ Cost = Total\ VM\ cost + Total\ Data\ transfer\ cost$$

VM and data transmission costs for these three techniques are identical, hence a cost analysis cannot be used to differentiate between them. Figure 5 represents the cost of the data center for VMs. The VMs cost starts at \$0.5 to \$3. The results show that at the 300 min for \$11.11, the cost for VMs has the highest rate among the others. The min-cost was 60 min is only \$0.5. Figure 6 shows the data transfer cost for the data center for 60 min to 300 min. The maximum rate for the data transfer cost is at time 300 min for \$8.61. The minimum data transfer cost rate is \$0.38 at a time of 60 min.

Figure 7 shows the response time by the region for the 6 userbases region in the CloudAnalyst simulation. The average speeds for the response time by region are the highest at UB4 at 500.83 ms. The maximum response time shown is 650.30 ms at a time of 300 min for region UB4. The minimum average response time is 50.12 ms at region UB1 for the time at 300 min.

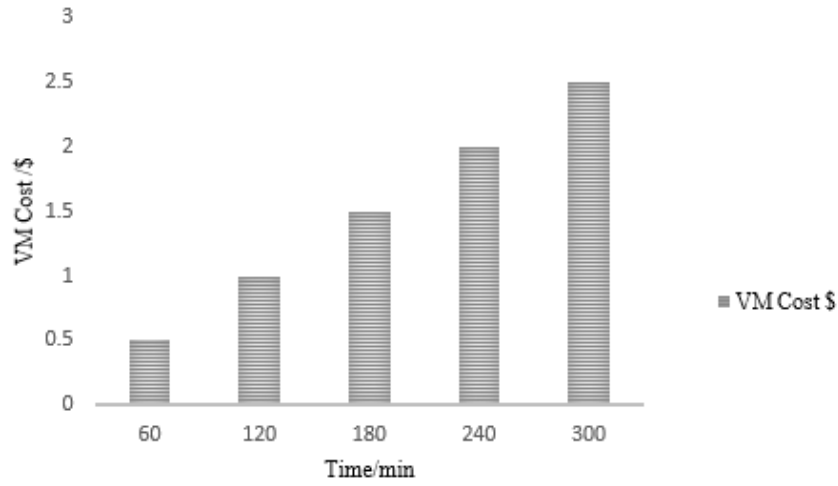


Figure 5. Analysis of the cost data center for VMs

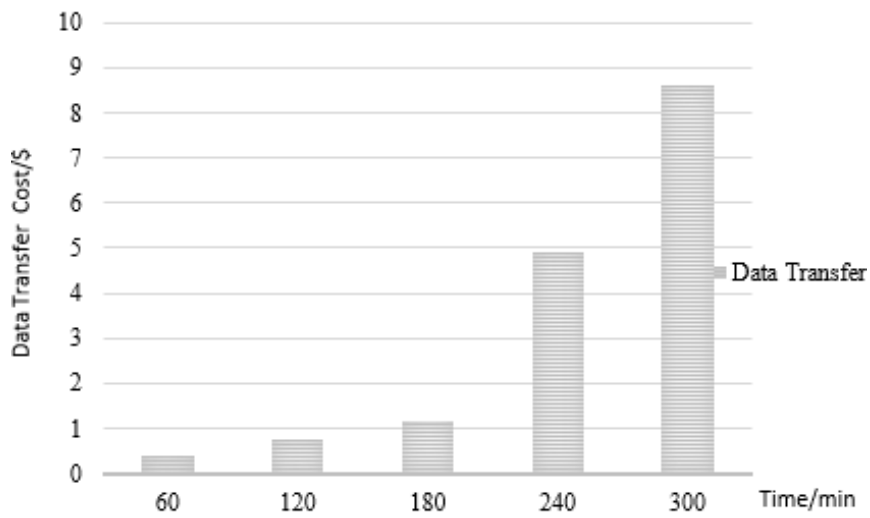


Figure 6. Analysis of data center transfer cost

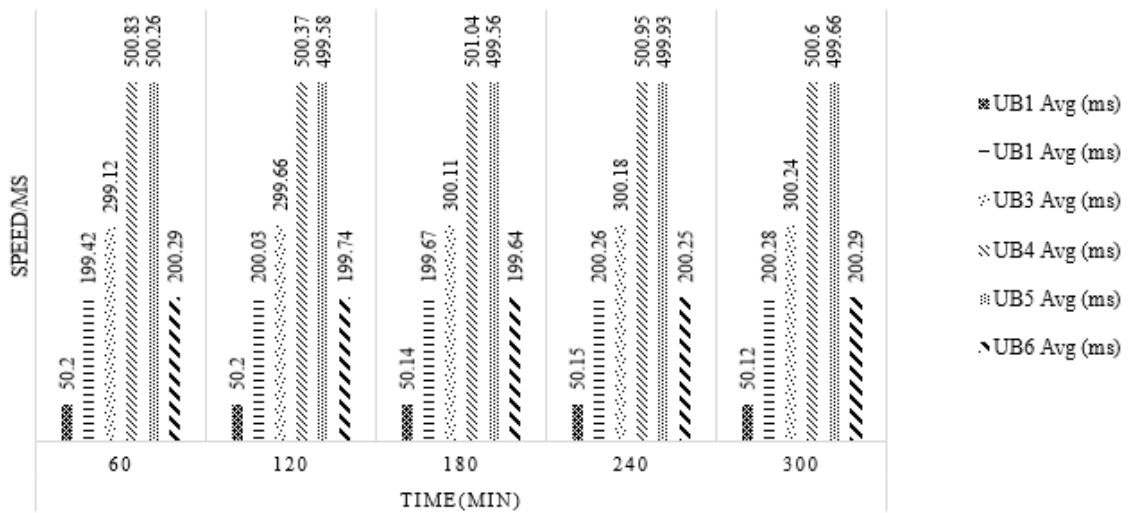


Figure 7. Analysis of response time by region

5. CONCLUSION

Increased demand for cloud services worldwide necessitates research into efficient load-balancing methods so that providers can keep up with customer demands and the evolving nature of the cloud. In this paper, we discussed the performance of the load balancing technique of RR along with different service broker policies, which this analysis is best to support the implementation of IoT services on cloud or fog computing networks. Analysis of the response time and cost analysis has been presented to confirm that the RR algorithms used in controlling resources are beneficial to the QoS and the implementation of IoT services by using online resources. The RR algorithm will be improved in the future by devising a novel technique for time quantum calculation that mixes dynamic and fixed quantum values to increase the RR algorithm's performance. Average waiting time and turnaround can be improved by adding input parameters such as job sequencing with a deadline or comparing other algorithms for job scheduling to find the best one that fits your needs.

ACKNOWLEDGEMENT

Authors would like to thank the Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Kompleks AI-Khawarizmi, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia for the support research and Research Management Centre (RMC), Universiti Teknologi MARA, UiTM Shah Alam for the support fund on grant no 600-RMC/GPM ST 5/3 (038/2021) in publishing this paper.




REFERENCES

- [1] L. Hamid, A. Jadoon, and H. Asghar, "Comparative analysis of task level heuristic scheduling algorithms in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 11, pp. 12931–12949, Jul. 2022, doi: 10.1007/s11227-022-04382-x.
- [2] A. M. O. Bandara and U. U. S. Rajapaksha, "An improved round robin algorithm for an efficient CPU scheduling," in *2022 2nd International Conference on Advanced Research in Computing (ICARC)*, Feb. 2022, pp. 349–354, doi: 10.1109/ICARC54489.2022.9754037.
- [3] A. R. Aswini and G. Prabaharan, "A scalable approach for efficient multicast data transmission in wireless networks," in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Feb. 2014, pp. 1–3, doi: 10.1109/ICICES.2014.7033854.
- [4] T. Balharith and F. Alhaidari, "Round robin scheduling algorithm in CPU and cloud computing: A review," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, May 2019, pp. 1–7, doi: 10.1109/CAIS.2019.8769534.
- [5] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," *Procedia Computer Science*, vol. 115, pp. 322–329, 2017, doi: 10.1016/j.procs.2017.09.141.
- [6] N. Srilatha, M. Sravani, and Y. Divya, "Optimal round robin CPU scheduling algorithm using Manhattan distance," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 6, pp. 3664–3668, Dec. 2017, doi: 10.11591/ijece.v7i6.pp3664-3668.
- [7] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 190–202, Feb. 2019, doi: 10.11591/ijece.v9i1.pp190-202.
- [8] H. Gibet Tani and C. El Amrani, "Cloud computing CPU allocation and scheduling algorithms using CloudSim simulator," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, no. 4, pp. 1866–1879, Aug. 2016, doi: 10.11591/ijece.v6i4.pp1866-1879.
- [9] N. S. Mohd Pakhrudin, M. Kassim, and A. Idris, "A review on orchestration distributed systems for IoT smart services in fog computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1812–1822, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1812-1822.
- [10] S. Kumar and A. Dumka, "Load balancing with the help of round robin and shortest job first scheduling algorithm in cloud computing," in *Proceedings of International Conference on Machine Intelligence and Data Science Applications*, 2021, pp. 213–223, doi: 10.1007/978-981-33-4087-9_19.
- [11] S. I. Suliman *et al.*, "An effective energy-efficient virtual machine placement using clonal selection algorithm," *International Journal of Advanced Technology and Engineering Exploration*, vol. 8, no. 75, pp. 412–421, Feb. 2021, doi: 10.19101/IJATEE.2020.762129.
- [12] S. Kaur and T. Sharma, "Efficient load balancing using improved central load balancing technique," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, Jan. 2018, pp. 1–5, doi: 10.1109/ICISC.2018.8398857.
- [13] V. Joshi, "Load balancing algorithms in cloud computing," *International Journal of Research in Engineering and Innovation*, vol. 3, pp. 530–532, 2019.
- [14] T. Deepa and D. Cheelu, "A comparative study of static and dynamic load balancing algorithms in cloud computing," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Aug. 2017, pp. 3375–3378, doi: 10.1109/ICECDS.2017.8390086.
- [15] A. I. El Karadawy, A. A. Mawgoud, and H. M. Rady, "An empirical analysis on load balancing and service broker techniques using Cloud analyst simulator," in *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, Feb. 2020, pp. 27–32, doi: 10.1109/ITCE48509.2020.9047753.
- [16] N. Thanh Long, "Research on applying hierarchical clustered based routing technique using artificial intelligence algorithms for quality of service of service based routing," *Internet of Things and Cloud Computing*, vol. 3, no. 3, 2015, doi: 10.11648/j.iotcc.s.2015030601.11.
- [17] Varadhaganapathy, "Authentication based and optimized routing technique in mobile ad hoc networks," *Journal of Computer Science*, vol. 7, no. 5, pp. 651–656, May 2011, doi: 10.3844/jcssp.2011.651.656.
- [18] A. K. Singh and M. Burhan, "Configuring dynamic capability architecture for understanding changes," *International Journal of Strategic Change Management*, vol. 7, no. 2, 2018, doi: 10.1504/IJSCM.2018.091598.




- [19] F. Alhaidari and T. Z. Balharith, "Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling," *Computers*, vol. 10, no. 5, May 2021, doi: 10.3390/computers10050063.
- [20] L. Zou, Z. Wang, Q.-L. Han, and D. Zhou, "Moving horizon estimation for networked time-delay systems under round-robin protocol," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 5191–5198, Dec. 2019, doi: 10.1109/TAC.2019.2910167.
- [21] S. Mostafa and H. Amano, "An adjustable round robin scheduling algorithm in interactive systems," *Information Engineering Express*, vol. 5, no. 1, 2019, doi: 10.52731/iee.v5.i1.353.
- [22] D. Biswas and M. Samsuddoha, "Determining proficient time quantum to improve the performance of round robin scheduling algorithm," *International Journal of Modern Education and Computer Science*, vol. 11, no. 10, pp. 33–40, Oct. 2019, doi: 10.5815/ijmeecs.2019.10.04.
- [23] A. A. Alsulami, Q. A. Al-Haija, M. I. Thanoon, and Q. Mao, "Performance evaluation of dynamic round robin algorithms for CPU scheduling," in *2019 SoutheastCon*, Apr. 2019, pp. 1–5, doi: 10.1109/SoutheastCon42311.2019.9020439.
- [24] X. Zongyu and W. Xingxuan, "A predictive modified round robin scheduling algorithm for web server clusters," in *2015 34th Chinese Control Conference (CCC)*, Jul. 2015, pp. 5804–5808, doi: 10.1109/ChiCC.2015.7260546.
- [25] S. Jeyalakshmi, J. Anita Smiles, D. Akila, D. Mukherjee, and A. J. Obaid, "Energy-efficient load balancing technique to optimize average response time and data center processing time in cloud computing environment," *Journal of Physics: Conference Series*, vol. 1963, no. 1, Jul. 2021, doi: 10.1088/1742-6596/1963/1/012145.
- [26] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "Cloud computing virtualization of resources allocation for distributed systems," *Journal of Applied Science and Technology Trends*, vol. 1, no. 3, pp. 98–105, Jun. 2020, doi: 10.38094/jastt1331.
- [27] N. Goel, A. Saxena, and G. R. B., "An optimal mathematical approach for CPU scheduling – OMDRRS vs. RR," *International Journal of Computational and Applied Mathematics*, vol. 12, no. 1, pp. 65–80, 2017.
- [28] D. P. Kumar, T. S. Reddy, and A. Y. Reddy, "Finding best time quantum for round robin scheduling algorithm to avoid frequent context switch," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 5, pp. 6750–6754, 2014.

BIOGRAPHIES OF AUTHORS






Nor Syazwani Mohd Pakhrudin    is a Ph.D. student in the School of Electrical Engineering, Universiti Teknologi MARA (UiTM), Shah Alam, Selangor Malaysia in Electrical Engineering Program. She received her master's degree in Telecommunication and Information Engineering, from Universiti Teknologi MARA (UiTM), Shah Alam, Selangor Malaysia under Faculty of Electrical Engineering in 2018. Previously, she obtained her first degree from Universiti Teknologi MARA (UiTM) Cawangan Pulau Pinang, with Honours, in electrical and electronic engineering. She has registered as a graduate engineer member with the Board of Engineers Malaysia (BEM) since 2018. She can be contacted at email: norsyazwanipakhrudin@gmail.com.



Murizah Kassim    is currently working at Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Centre of Excellence, Universiti Teknologi MARA, UiTM Shah Alam. She is an Associate Professor from the School of Electrical Engineering, College of Engineering, UiTM Shah Alam, Selangor. She received her Ph.D. in Electronic, Electrical and System Engineering in 2016 from the Faculty of Built Environment and Engineering, Universiti Kebangsaan Malaysia (UKM). She has published about 118 indexed papers related to computer networks, IoT, and Web and Mobile development applications research. She has 19 years of experience in the technical team at the Centre for Integrated Information System, UiTM. She is also a member of Enabling Internet of Things Technologies (Eliott) research group UiTM. She joined the academic in January 2009 and is currently a member of MBOT, IEEE, IET, IAENG, and IACSIT organizations. She can be contacted at murizah@uitm.edu.my.



Azlina Idris    is an associate professor at the Universiti Teknologi MARA (UiTM), Selangor, Malaysia. She obtained her Ph.D. in wireless communication from Universiti Malaya (UM), Malaysia. She has received the Master of Engineering in Electrical from Universiti Teknologi Malaysia (UTM). Previously, she obtained her first degree from Leeds Metropolitan University, United Kingdom with Honours, in applied computers. She is a member of the Wireless Communication Technology (WiCOT) Research Interest Group (RIG) and her research interests include OFDM/OFDMA transmission, single and multiuser precoding, modulation, MIMO transmission techniques and receivers, channel coding, interference management and mitigation, and channel modeling (channel estimation). She can be contacted at She can be contacted at email: azlina831@uitm.edu.my.