An ameliorated Round Robin algorithm in the cloud computing for task scheduling

Article Info

Keywords:

An ameliorated Round Robin algorithm
Average turnaround time
Average waiting time
Cloud computing
Round Robin
Task scheduling

ABSTRACT

Cloud computing is an advanced technology that offers types of assistance on requests. Because of the huge measure of requests got from cloud clients, all requests should be managed efficiently. Therefore, the task scheduling is critical in cloud computing. The provision of computational resources in cloud is controlled by a cloud provider. It is necessary to design high-efficiency scheduling algorithms that are compatible with the corresponding computing paradigms. This paper introduces a new task scheduling method for cloud computing called an ameliorated Round Robin algorithm (ARRA). The proposed algorithm develops an optimal time quantum based on the average of task burst time using fixed and dynamic manners. The experimental results showed that the ARRA significantly outperformed other algorithms including improved RR, enhanced RR, dynamic time quantum approach (ARR) and enhanced RR (RAST ERR) in terms of the average waiting time, average turnaround time and response time.

1. INTRODUCTION

Cloud computing is a new computing technology that represents a significant step forward in the development and deployment of a growing variety of applications. Cloud computing is a model built on the use of clouds. The cloud is a collection of software and hardware that work together to provide various aspects of computing to the end-user as online services [1]. Cloud computing applications may be accessed anytime and from anywhere [2]. Many popular services and websites are hosted in the cloud [3], [4]. Task scheduling is one of the main issues with cloud computing [5]. It means using an efficient algorithm to map the tasks of the clients to the available and appropriate resources [6], [7]. Scheduling n tasks on m resources is characterized as an NP-hard problem with O (mn) run time complexity [8]. So it is important to use an effective task scheduling algorithm to enhance the performance of the system [8]. In a cloud computing system, three different task scheduling techniques are available: i) traditional algorithms [9], [10], like Round Robin (RR) [11], ii) heuristic algorithms, like (MCT), (MET) [12] and max-min [13], and iii) meta-heuristic algorithms, like ACO [14], PSO [15] and GA [16]. The RR algorithm is a popular scheduling algorithm in the cloud computing [17], [18].

The RR algorithm [19]: it is a preemptive algorithm [10]. It works well in time sharing [20]. For interactive users, these environments must ensure reasonable response times. In the RR scheduling algorithm, every process in the ready queue receives a time slice (time quantum (TQ)) [21]. The current process is placed at the end of this ready queue when the TQ expires. RR decreases the average turnaround time (ATT)

and average waiting time (AWT) [6]. The most significant issue with the RR method is the TQ length [19], [22]. Setting a TQ too short causes too many context switches, which lowers CPU efficiency. And on the other hand, if the TQ is set too long, the algorithm trends toward the FCFS algorithm and may result in a slow response time. So, it is important to enhance the RR algorithm to reduce AWT, ATT and response time.

The main contribution of this paper is to improve the RR algorithm by enhancing the TQ in cloud computing for solving task scheduling problem. A novel algorithm for changing TQ in a progressive manner at various states of the ready queue is proposed. A mathematical model has been created to prove that the proposed algorithm outperforms the traditional RR algorithm in terms of several performance metrics such as AWT, ATT and response time (ART). According to the experimental results, the proposed improved version of the RR algorithm outperforms the traditional RR algorithm. This proposed algorithm solves the problem by using a progressive TQ, which is based on the average task burst time using fixed and dynamic manners. Furthermore, the processes are sorted in ascending order of their burst time, and then the proposed algorithm is applied to each process to improve turnaround time, waiting time and response time. In comparison to the other RR techniques discussed in this work, the drawbacks of the discussed algorithms like Improved RR [23], enhanced RR [24], ARR [25] and enhanced RR (RAST ERR) [26] are that they give a large AWT, ATT and ART. The contribution of this work is to: i) minimize AWT, ii) minimize ATT, iii) minimize response time. This paper is divided into six sections. The related work is in section 2, the proposed algorithm in section 3, the method in section 4 and the results in section 5. The conclusion in section 6.

2. RELATED WORK

According to Sangwan *et al.* [23], an improved RR algorithm is proposed. At the arrival of the CPU's arbitrary requests, the method calculates the mean of the burst time. The TQ is the mean of the burst times. The first process in the queue is selected, and the CPU is allocated for the chosen TQ. Then the remaining burst times are moved to the tail of the ready queue [21]. Sometimes the first process is too large and the rest processes are too small, so the response time decreases which causes an increase in waiting time and turnaround time. Research by Mittal *et al.* [24], an enhanced RR algorithm is offered. It consists of two other existing algorithms, namely (resource allocation scheduling algorithm) RASA and (shortest job first) SJF. SJF coordinates all procedures in the prepared line to minimize burst time, and RASA aids in preventing system starvation and reducing response time for larger tasks. The TQ is updated after each cycle using the same hybrid algorithm.

According to Pradhan *et al.* [4], a modified RR algorithm is suggested. This algorithm starts with a time equal to the burst time of the first request, which changes once it is completed. The algorithm calculates the average of the requests waiting in the ready queue, including the newly arrived request. The disadvantage of this algorithm is that if the first processes are too small, it will cause many context switches and therefore will increase the AWT and ATT. According to Fataniya and Patel [25], a dynamic time quantum approach (ARR) to improve the RR is presented here. It is based on the MRRA [4] and SRBRR [27] algorithms. It has been offered to improve cloud computing resource allocation. It is a dynamic RR in which the TQ is always changing. TQ is calculated as the sum of the mean and median divided by two for each round. The disadvantage of this algorithm is that if the first processes are too small, it will cause many context switches and therefore will increase the average waiting time and ATT. Mora *et al.* [28] offered another RR algorithm called modified median RR algorithm (MMRRA). The TQ is dynamically allocated by determining a modified median of tasks. Mayuree *et al.* [28] introduced a RR based on remaining time and a median algorithm (RR_RT&M). Based on remaining time and task median, the TQ is modified. If the number of tasks is less than or equal to 3, the TQ is the maximum remaining time of task. Otherwise, TQ is the median of task time [29].

According to Tani and Amrani [30], a variant on RR (VORR), which is one of the improvements of the RR algorithm. It effectively exploits the CPU by setting up an effective TQ based on the median of burst times. Hicham *et al.* [30] introduced a smarter SRR algorithm which uses the concept of RR but the TQ is changed dynamically depending on the number of tasks in a queue [31]. The SJF algorithm is used to distribute the time if there are fewer than or equal to three tasks. However, it utilizes the average of the burst time of tasks if the number of tasks is larger than 3 and even. Otherwise, it uses the median of task burst time instead. According to Stephen *et al.* [26], an enhanced RR algorithm (RAST ERR technique) that has been suggested uses mean for dynamic time quantum. Burst time (the amount of time it takes to complete a task) is used to compute the mean for the provided tasks and is then set as the time quantum. For the first iteration, initial burst time and mean are used. The finished tasks will be deleted from the queue after the initial iteration; otherwise, the next iteration will begin. The remaining burst time of the second iteration is used to calculate the mean value and the remaining burst time for each task will determine how long it takes to complete. Then, in the following iteration, the mean is once more determined using the remaining burst time

for task and set as the time quantum. Until there are no tasks waiting in the queue to be completed, the procedure of obtaining the mean and establishing the time quantum is repeated. The waiting time and task turnaround time of the suggested method are distinguishing features.

All of the previous RR CPU scheduling enhancements have some drawbacks. The system's incoming processes might have different burst times, which mean that their CPU execution times might also change. The turnaround time and waiting time can be reduced if all of the processes are sent to the CPU for execution in ascending order. The RR algorithm uses a fixed time quantum to operate (TQ). There are two results for the RR algorithm: either time quantum is high or low. The RR method will operate on a first-come, first-served (FCFS) basis if the time quantum is large. If the time quantum is small, the algorithm will fail and result in a significant number of context switches. So, this paper proposes an optimal time quantum that solves this problem, which uses fixed and dynamic manners and enhances the performance of the system by: maximizing CPU utilization, minimizing waiting time, turnaround time and response time. Many algorithms have been developed for enhancing RR algorithm such as improved RR [23], enhanced RR [24], ARR [25] and enhanced RR (RAST ERR) [26]. However, these algorithms still have higher AWT, ATT and response time. The main focus of this work is the suggestion of a novel algorithm called an ameliorated RR algorithm (ARRA) in cloud computing that minimizes the drawbacks of the RR algorithm by raising the performance metrics by decreasing the AWT, ATT and response time for some algorithms. This is done by choosing an optimal TQ that achieves low waiting time, turnaround time and response time. Thus, it ends starvation. If tasks arrive at the same time, it uses a fixed TQ and a dynamic TQ if tasks arrive at different times.

3. THE PROPOSED ALGORITHM

This study aims to improve system efficiency by introducing an ameliorated RR scheduling algorithm by minimizing metrics like waiting time, turnaround time and response time in the cloud. To optimize the task scheduling process, the intended algorithm has put the main focus on computing the time quantum effectively. The ARRA uses both fixed and dynamic manners according to the task arrivals. When the tasks arrive at the same time, a fixed TQ is applied. On the other hand, when the tasks arrive at different times, a dynamic TQ is applied. All the processes that are present in the ready queue are arranged in an increasing order. When two or more processes occur in the ready queue with the same burst time, they are rearranged in the ready queue according to when they occurred. The average is then computed for all of the tasks in the ready queue. After that, the TQ is calculated according to (1) as determined by [32]:

$$TQ_i = \frac{3}{4} \times \frac{\sum_{i=1}^{n} BT_i}{n} \quad (1)$$

Where TQ_i is the time quantum for tasks, BT_i is the burst time for task i and n is the number of tasks. At the beginning, the algorithm allocates system resources to the first task in the ready queue. When the task process or TQ length is finished, the associated mechanism verifies the statutes of the task. If the remaining burst time for the currently executed task is less than half of the current set time quantum, the algorithm allocates the CPU to the same task; otherwise, the task is put to the back of the queue. With minimal average waiting and turnaround times, the newly implemented algorithm effectively enhances task scheduling. The pseudo code for the ARRA is given below. The flowchart of the ARRA algorithm is shown in Figure 1.

1- Tasks arrive in the cloud.
2- Load distribution: load the processes in the datacenter using circular load distribution algorithm.
   For all datacenters,
3- If (ready queue! =0), then arrange all the tasks in an increasing order by their burst time in the ready queue.
4- Calculate the average of all the burst times.
5- Consider the (TQ) to be equal to ¾ average of burst times.
6- Assign this TQ to all tasks inserted in the ready queue.
7- If the current task has a burst time less than half of the time quantum, then allocate it again.
   Else
8- The remaining part of the current task is moved to the back of the ready queue.
9- Repeat steps 3-7 for all the tasks until the TQ expires for each one.
10- If a new (task) arrives, then update the counter and go to step 4.
11- Calculate the (AWT), (ATT) and average response time.
12- Stop and exit.

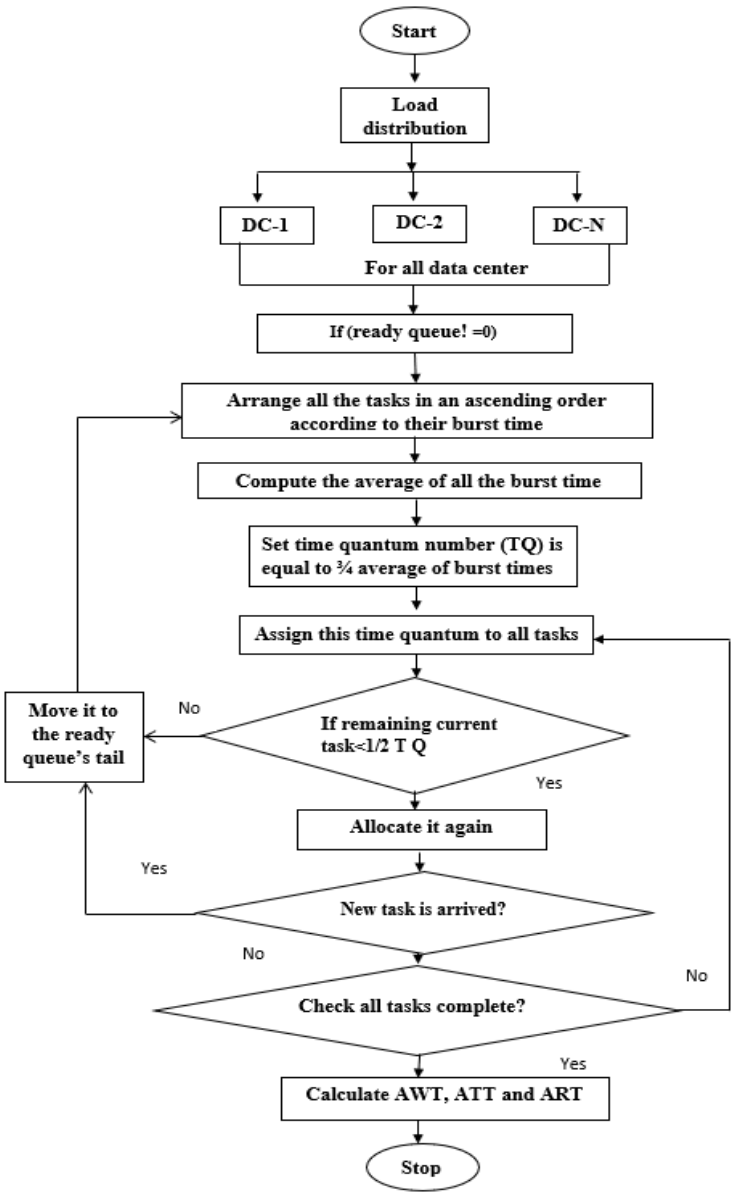

Figure 1. Flowchart of the ARRA algorithm

4. METHOD

In cloud computing, users send multiple requests at the same time. The RR task scheduling algorithm works on both different and same arriving times. To evaluate the effectiveness of the proposed algorithm, various tasks are applied to RR [19], improved RR [23], enhanced RR [24], ARR [25], enhanced RR (RAST ERR) [26] and the ARRA algorithms. A comparative study among these algorithms is presented here. The comparison criteria is restricted to AWT, ATT and response time.
The AWT is calculated as (2):

$$AWT=\Sigma\ (TAT\ of\ P_i - BT\ of\ P_i) / N \quad (2)$$

The ATT is calculated as (3):

$$ATT=\Sigma\ (T_i - AT\ of\ P_i) / N \quad (3)$$

Where *ATT* of P_i is the turnaround time of task P_i, *BT* of P_i is the burst time of task P_i, T_i is the exit time for task, *AT* of P_i is the arrival time of task and N is the number of all tasks. To test the effectiveness of the proposed algorithm, two scenarios are used.

4.1. Case 1 (zero arrival time)

In case 1, seven tasks which are T1, T2, T3, T4, T5, T6 and T7 are integrated with CPU burst time and zero arrival time. These values of burst time are shown in Table 1. Figures 2-7 shows of the RR, improved RR, enhanced RR, ARR, enhanced RR (RAST ERR) and ARRA.

Table 1. The burst time for tasks

Tasks	Burst time(ms)
T1	105
T2	60
T3	120
T4	48
T5	75
T6	160
T7	145

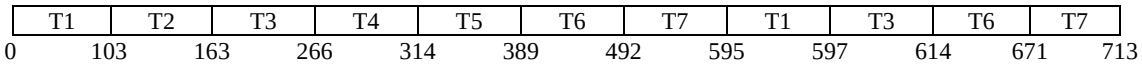

Figure 2. Gantt chart for RR algorithm [19]

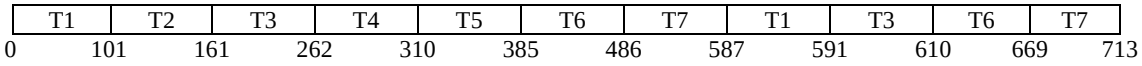

Figure 3. Gantt chart for improved RR algorithm [23]

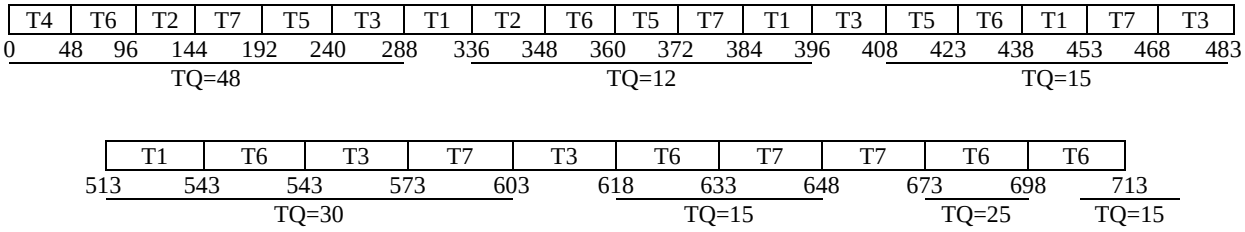

Figure 4. Gantt chart for the enhanced RR algorithm [24]

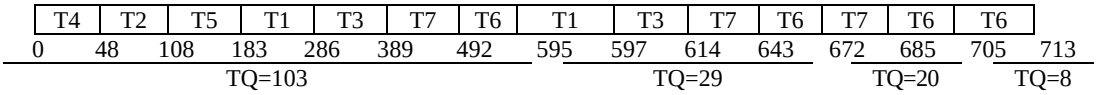

Figure 5. Gantt chart for ARR algorithm [25]

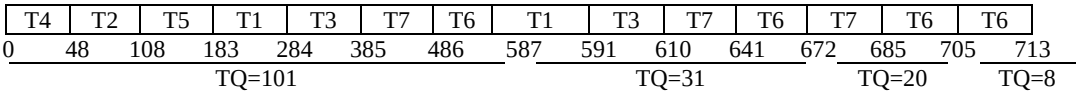

Figure 6. Gantt chart for enhanced RR (RAST ERR) algorithm [26]

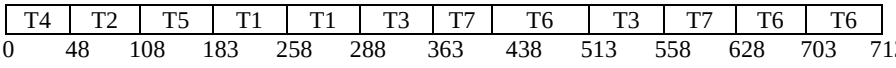

Figure 7. Gantt chart for ARRA algorithm

The turn-around time, waiting time and response time of the ARRA algorithm for the above example are calculated as shown in Table 2. A comparative study among the RR, Improved RR, Enhanced RR, ARR, Enhanced RR (RAST ERR) and ARRA algorithms with respect to TQ, AWT, ATT and average response time (ART) as shown in Table 3. Figure 8 compares the performance of the preceding algorithms using AWT, ATT and ART.

Table 2. T.A.T and W.T for ARRA algorithm

Tasks	WT	TAT	Response time
T1	183	288	183
T2	48	108	48
T3	438	558	288
T4	0	48	0
T5	108	183	108
T6	553	713	438
T7	483	628	363

Table 3. Comparative study of RR, improved RR, enhanced RR, ARR, enhanced RR (RAST ERR) and ARRA algorithms (case 1)

Algorithm	TQ	AWT	ATT	ART
RR	103	392.57	494.42	246.7
Improved RR	101	389.42	491.28	243.5
Enhanced RR	48,12,15,30, 15,25,15	374.71	476.57	144
ARR	103,29,20,8	319.28	421.14	215.1
Enhanced RR (RAST ERR)	101,31,20,8	317.85	419.71	213.4
ARRA	75	259	360.85	204

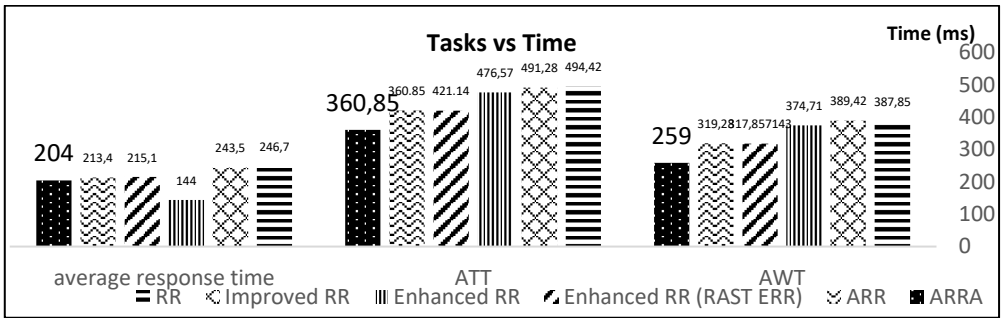

Figure 8. Comparison graph for AWT, ATT and ART for (case 1)

4.2. Case 2(non-zero arrival time)

In case 2, the tasks have non-zero arrival times in the ready queue, so a dynamic TQ is applied for the proposed algorithm in this case. TQ is updated each round. Five tasks, which are T1, T2, T3, T4 and T5 are integrated with CPU burst time with non-zero arrival times as shown in Table 4. Figures 9-14 shows of the RR, improved RR, enhanced RR, ARR, enhanced RR (RAST ERR) and ARRA algorithms.

Table 4. The arrival time and burst time for tasks

Tasks	Arrival time(ms)	Burst time(ms)
T1	0	40
T2	5	25
T3	10	60
T4	15	100
T5	20	75

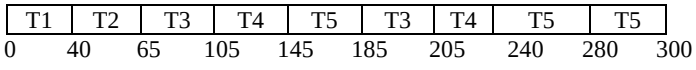

Figure 9. Gantt chart for RR algorithm [19]

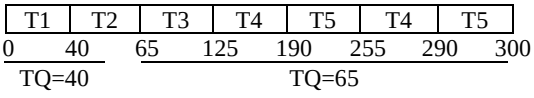

Figure 10. Gantt chart for improved RR algorithm [23]

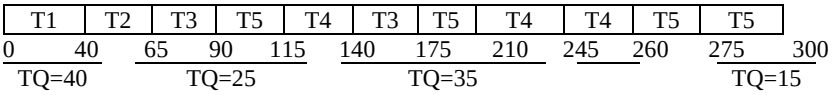

Figure 11. Gantt chart for enhanced RR algorithm [24]

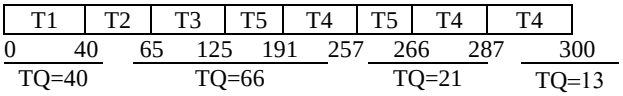

Figure 12. Gantt chart for ARR algorithm [25]

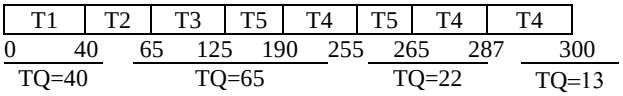

Figure 13. Gantt chart for enhanced RR (RAST ERR) algorithm [26]

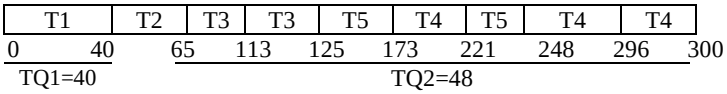

Figure 14. Gantt chart for ARRA algorithm

Turnaround time, waiting time and response time for above example of the ARRA algorithm is calculated as shown in Table 5. Table 6 presents a comparative study among the RR, improved RR, enhanced RR, ARR, enhanced RR (RAST ERR) and ARRA algorithms with respect to TQ, AWT, ATT and ART. Figure 15 shows the comparison of AWT, ATT and ART for the previous algorithms in (case 2).

Table 5. T.A.T, W.T and response time for ARRA algorithm

Tasks	WT	TAT	Response time
T1	0	40	0
T2	35	60	40
T3	135	195	65
T4	125	225	105
T5	205	280	145

Table 6. Comparative study of RR, enhanced RR, ARR, enhanced RR (RAST ERR) and ARRA algorithms (case 2)

Algorithm	TQ	AWT	ATT	ART
RR	40	100	160	71
Improved RR	40,65	94	154	84
Enhanced RR	40,25,35,15	98	158	62
ARR	40,66,21,13	89.2	149.2	84.2
Enhanced RR (RAST ERR)	40,65,22,13	89	149	84
ARRA	40, 48	85.6	145.6	80.6

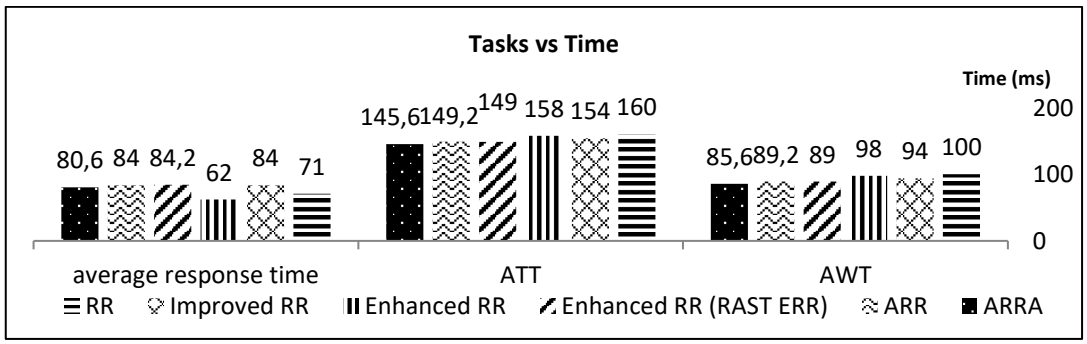

Figure 15. Comparison graph for AWT, ATT and ART (case 2)

4.3. Percentage of improvements

The percentage of each performance metric improvement [29] when compared to ARRA is in (4):

$$Improvement\ percentage=\frac{(X_{other})-(X_{proposed})}{X_{other}} \times 100 \quad (4)$$

Where *x* is a performance metric that is AWT or ATT or ART and *other* refers to other algorithms that are used to compare with ARRA. The scheduling algorithm will be improved if the AWT, ATT and ART are lower. Table 7 and Table 8 show the improvement percentage for each metric for all algorithms compared with the ARRA algorithm in case 1 and case 2. The percentage of improvement for AWT ranges between 3.8-34.02%. Similarly, the improvement percentage of ATT varies between 7.84-27.01% and the ART varies between 0-17.3% as shown in Table 9. Therefore, compared to other algorithms, the ARRA performs the best in case 1 and case 2.

Table 7. Percentage of improvement for AWT comparing with ARRA algorithm

	RR (%)	Improved RR (%)	Enhanced RR (%)	ARR (%)	Enhanced RR (RAST ERR) (%)
Case 1	34.02	33.49	30.87	18.87	22.72
Case 2	14.4	4.04	12.6	4.03	3.82

Table 8. Percentage of improvement for ATT comparing with ARRA algorithm

	RR (%)	Improved RR (%)	Enhanced RR (%)	ARR (%)	Enhanced RR (RAST ERR) (%)
Case 1	27.01	26.54	24.28	14.31	14.02
Case 2	9	2.28	7.84	2.41	2.82

Table 9. Percentage of improvement for ART comparing with ARRA algorithm

	RR (%)	Improved RR (%)	Enhanced RR (%)	ARR (%)	Enhanced RR (RAST ERR) (%)
Case 1	17.3	16.2	0	5.1	4.4
Case 2	0	2.28	0	4.27	4.04

4.4. Simulation

To evaluate the effectiveness of the proposed model, the benchmark was simulated using the same parameters while using various algorithms, considering the burst time and task arrival time. A simulation is implemented in C++ and made for the proposed algorithm(ARRA), improved RR [23], enhanced RR [24], ARR [25], enhanced RR (RAST ERR) [26] and all of these algorithms are compared with the RR algorithm in order to evaluate their performance. The model is a single resource in the form of virtual machines with the same random data set. The comparison of these algorithms is based on AWT, ATT and ART. Because the number of tasks in the ready queue determines average waiting and turnaround time, an increase in time results in a rise in cost.

5. RESULTS AND DISCUSSION

The experimental data used varied data sets from (2000-8000) tasks. The values of burst time tasks are generated randomly in the range of 1–100, and all tasks arrive at the same time. Several experiments were carried out to validate the proposed model, which is repeated numerous times while the number of tasks is increasing. The comparison of algorithms in terms of AWT is shown in Figure 16. For tasks (2000 to 8000), the stacked line chart is plotted. The AWT of the tasks is provided in milliseconds and plotted by the y-axis, while the number of tasks in the ready queue is plotted by the x-axis. The proposed algorithm (ARRA) gives better results, followed by enhanced RR (RAST ERR) [26], ARR [25], enhanced RR [24] and improved RR [23]. A substantial improvement is given by these algorithms when compared to the RR algorithm. As the number of tasks increases in the ready queue, the performance of the algorithms is enhanced. When compared to RR, the enhanced RR (RAST ERR) [26], ARR [25], and enhanced RR [24] produce significant results, while the improved RR [23] produces reasonable improvement results. Whereas the proposed algorithm shows more significant improvement results than other algorithms. While inceasing in number of tasks, the performance of ARRA showed an upward trend in AWT compared to other algorithms. In comparison to suggested algorithms, the AWT for RR is consistently increasing, as seen in the line chart.

The behaviour of algorithms in terms of ATT exhibits a similar pattern to that shown in Figure 17. For tasks (2000 to 8000), the stacked line chart is plotted. The ATT of the tasks is provided in milliseconds and plotted by the y-axis, while the number of tasks in the ready queue is plotted by the x-axis. The proposed

algorithm (ARRA) gives better results, followed by enhanced RR (RAST ERR) [26], ARR [25], enhanced RR [24] and improved RR [23]. A substantial improvement is given by these algorithms when compared to the RR algorithm. As the number of tasks in the ready queue increases, the performance of the algorithms is enhanced. When compared to RR, the enhanced RR (RAST ERR), ARR, and enhanced RR produce significant results, while the improved RR produces reasonable improvement results. The proposed algorithms act similarly, however while increasing in number of tasks, the performance of ARRA showed an upward trend in ATT compared to other algorithms. In comparison to suggested algorithms, the ATT for RR is consistently increasing, as seen in the line chart.

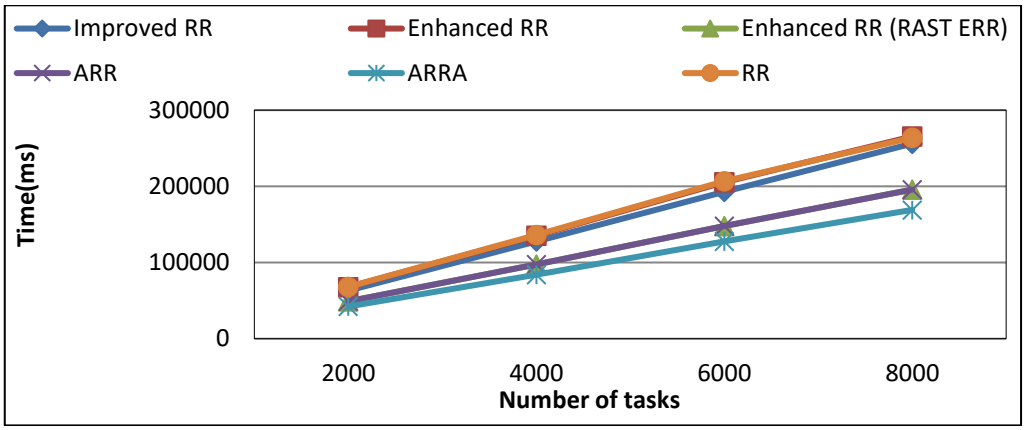

Figure 16. Comparative graph of AWT

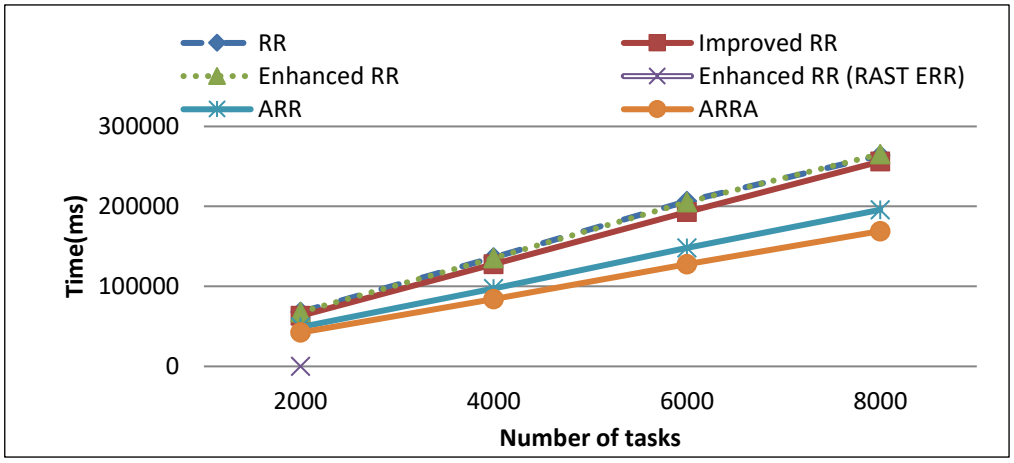

Figure 17. Comparative graph for ATT

The comparison of algorithms in terms of ART is shown in Figure 18. For tasks (2000 to 8000), the stacked line chart is plotted. The ART of the tasks is provided in milliseconds and plotted by the y-axis, while the number of tasks in the ready queue is plotted by the x-axis. The proposed algorithm (ARRA) gives better results than Enhanced RR (RAST ERR), ARR, and Improved RR. A substantial improvement is given by these algorithms when compared to the RR algorithm. As the number of tasks increases in the ready queue, the performance of the algorithms is enhanced. When compared to RR, the enhanced RR (RAST ERR), ARR and improved RR produce significant results, while the enhanced RR gives better results than the proposed algorithm. Whereas the proposed algorithm shows more significant improvement results than other algorithms. While inceasing in number of tasks, the performance of ARRA showed an upward trend in ART compared to other algorithms.Table 10 shows the high percentage of improvement for the ARRA compared with the benchmarked algorithms, where the proposed algorithm enhanced the AWT by 13.61-38.20% and the ATT by 13.61-38.19%. The ART is enhanced by 0-28%.

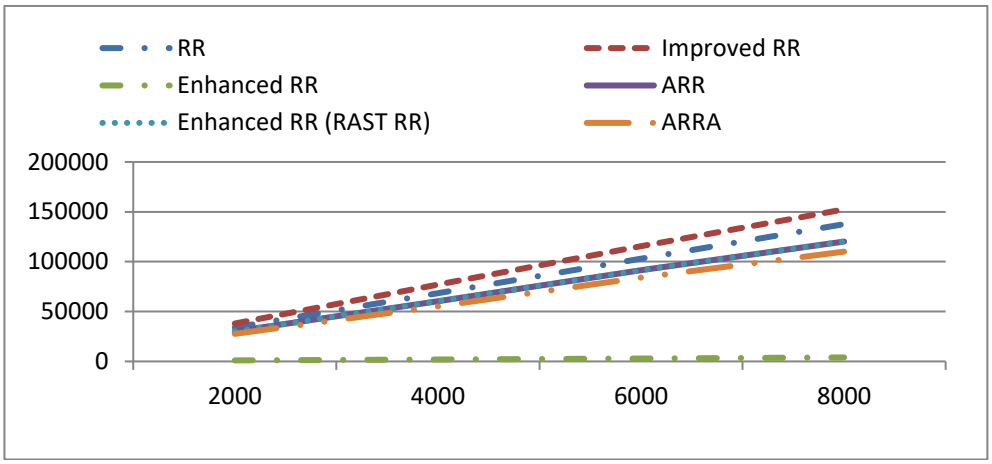

Figure 18. Comparative graph for ART

Table 10. Percentage of improvement for AWT, ATT and ART comparing with ARRA

Number of tasks	RR (%)			Improved RR (%)			Enhanced RR (%)			ARR (%)			Enhanced RR (RAST ERR) (%)		
	AW	AT	AR	AW	AT	AR	AW	AT	AR	AW	AT	AR	AW	AT	AR
N=200	37.8	37.8	18.8	33.0	33.0	27.3	37.6	37.6	0	13.8	13.8	8.3	13.88	13.8	8.3
N=400	38.2	38.1	19.2	34.1	34.0	28	37.8	37.8	0	13.6	13.6	8.1	13.62	13.6	8.1
N=600	38.1	38.1	18.3	33.8	33.8	27.3	37.8	37.8	0	13.7	13.6	8.1	13.70	13.6	8.1
N=800	35.9	35.8	20	33.9	33.9	27.7	36.3	36.2	0	13.6	13.6	8.4	13.61	13.6	8.4

5.1. Why choosing TQ=(3/4)* average?

Different ratios of average have been studied for the TQ to determine which ratio is the best. Different random datasets have been taken; dataset 1, dataset 2 and dataset 3 where the number of tasks n=4000, 10 and 5 tasks, with random burst times in the range (1 to 100). When taking ratios from TQ=(0.85*average) to TQ=(1.25*average) this gives non trusted values, which might either produce high or low AWT and ATT, or they could produce stable results with the same value as shown in Figures 19-20. Figures 19-20 show that the resulting AWT is stable and has the same value from TQ=(1.35*average) to TQ=1.95*average. It is obvious above that the results.

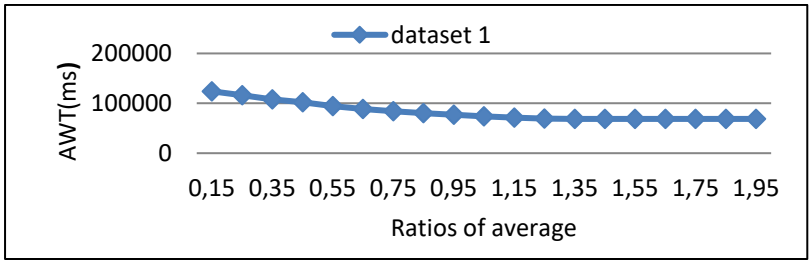

Figure 19. Comparison of several average ratios in dataset 1

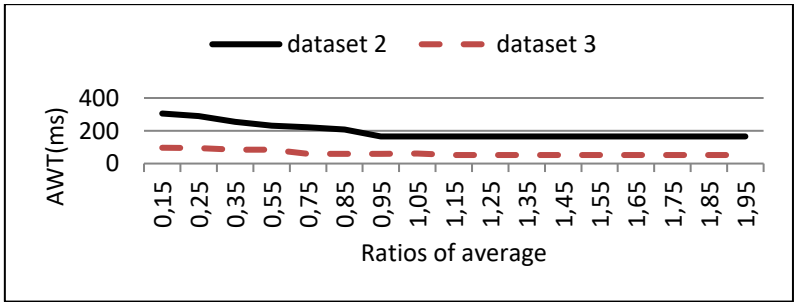

Figure 20. Comparison of several average ratios in dataset 1 and dataset 2

Given from TQ=(1.5*average) to TQ=(1.65*average) are decreased, and the results from TQ=(1.85*average) to TQ=(1.95 *average) are non-trusted values or have the same value. So whether the burst times are large or small, TQ=(0.75* average) is the ideal time quantum as shown in Figure 20.

6. CONCLUSION

Task scheduling is so important in cloud computing. In this work, an ameliorated scheduling algorithm (ARRA) has been proposed to enhance the performance metrics; AWT, ATT and ART in cloud computing. For tasks that come at the same time and at different times, the TQ is determined. The proposed ARRA algorithm showed that (0.75*average) is the ideal time quantum that should be allocated to the tasks in increasing order. It is simulated and compared with the RR, Improved RR, Enhanced RR, ARR and Enhanced RR (RAST ERR) algorithms. From the experiments, the results showed that the ARRA algorithm has considerably reduced the AWT by 3.8-38.20% and the ATT by 2.28-38.19% compared to other algorithms. In future work, the proposed algorithm can be modified to conclude other criteria.

REFERENCES

[1] A. Sharma and A. Sahu, *A Modified Round Robin Method to Enhance the Performance in Cloud Computing*. Springer Singapore, 2021, doi: 10.1007/978-981-15-5243-4_49.

[2] A. Razaque, N. R. Vennapusa, N. Soni, G. S. Janapati, and K. R. Vangala, "Task scheduling in Cloud computing," *2016 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2016*, pp. 1–5, 2016, doi: 10.1109/LISAT.2016.7494149.

[3] S. Banerjee, M. Adhikari, and U. Biswas, "Design and analysis of an efficient QoS improvement policy in cloud computing," *Service Oriented Computing and Applications*, vol. 11, no. 1, pp. 65–73, 2017, doi: 10.1007/s11761-016-0196-3.

[4] P. Pradhan, P. K. Behera, and B. N. B. Ray, "Modified Round Robin Algorithm for Resource Allocation in Cloud Computing," *Procedia Computer Science*, vol. 85, no. Cms, pp. 878–890, 2016, doi: 10.1016/j.procs.2016.05.278.

[5] B. Taneja, "An empirical study of most fit, max-min and priority task scheduling algorithms in cloud computing," *International Conference on Computing, Communication and Automation, ICCCA 2015*, pp. 664–667, 2015, doi: 10.1109/CCAA.2015.7148457.

[6] F. Alhaidari and T. Z. Balharith, "Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling," *Computers*, vol. 10, no. 5, 2021, doi: 10.3390/computers10050063.

[7] F. Alhaidari, T. Balharith, and E. Al-Yahyan, "Comparative analysis for task scheduling algorithms on cloud computing," *2019 International Conference on Computer and Information Sciences, ICCIS 2019*, pp. 1–6, 2019, doi: 10.1109/ICCISci.2019.8716470.

[8] R. Abu Khurma, H. Al Harahsheh, and A. Sharieh, "Task scheduling algorithm in cloud computing based on modified round robin algorithm," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 17, pp. 5869–5888, 2018.

[9] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS ONE*, vol. 12, no. 5, pp. 1–26, 2017, doi: 10.1371/journal.pone.0176321.

[10] M. Kumar Mishra and F. Rashid, "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum," *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 4, pp. 1–8, 2014, doi: 10.5121/ijcsea.2014.4401.

[11] K. Eldahshan, A. Abdelkader, and N. Ghazy, "Round Robin based Scheduling Algorithms, A Comparative Study," *Automatic Control and System Engineering Journal*, vol. 17, no. 2, pp. 29–42, 2017.

[12] I. A. Thiyeb and Dr. S. A. Alhomdy, "HAMM A Hybrid Algorithm of Min-Min and Max-Min Task Scheduling Algorithms in Cloud Computing," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 4, pp. 209–218, 2020, doi: 10.35940/ijrte.d4874.119420.

[13] S. Devipriya and C. Ramesh, "Improved max-min heuristic model for task scheduling in cloud," *Proceedings of the 2013 International Conference on Green Computing, Communication and Conservation of Energy, ICGCE 2013*, pp. 883–888, 2013.

[14] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," *Proceedings - 2013 8th International Conference on Computer Engineering and Systems, ICCES 2013*, vol. 12, no. 2, pp. 64–69, 2013, doi: 10.1109/ICCES.2013.6707172.

[15] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A Survey of PSO-Based Scheduling Algorithms in Cloud Computing," *Journal of Network and Systems Management*, vol. 25, no. 1, pp. 122–158, 2017, doi: 10.1007/s10922-016-9385-9.

[16] F. A. Emara, A. A. A. Gad-Elrab, A. Sobhi, and K. R. Raslan, "Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 5, pp. 571–582, 2021.

[17] T. Balharith and F. Alhaidari, "Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review," *2nd International Conference on Computer Applications and Information Security, ICCAIS 2019*, pp. 1–7, 2019, doi: 10.1109/CAIS.2019.8769534.

[18] S. Tayal, "Tasks scheduling optimization for the cloud computing systems," *International journal of advanced engineering sciences and technologies*, vol. 5, no. 2, pp. 111-115., 2011.

[19] O. H. M. Dorgham and M. O. Nassar, "Improved Round Robin Algorithm: Proposed Method to Apply {SJF} using Geometric Mean," *International Journal of Advanced Studies in Computers, Science and Engineering*, vol. 5, no. 11, pp. 112–119, 2016.

[20] P. Banerjee, B. Kumar, and P. Banerjee, "Mixed Round Robin Scheduling for Real Time Systems," *International Journal of Computer Trends and Technology*, vol. 49, no. 3, pp. 189–195, 2017, doi: 10.14445/22312803/ijctt-v49p130.

[21] S. Z. Iqbal *et al.*, "Relative time quantum-based enhancements in Round robin scheduling," *Computer Systems Science and Engineering*, vol. 41, no. 2, pp. 461–477, 2022, doi: 10.32604/csse.2022.017003.

[22] S. Banerjee, A. Chowdhury, S. Mukherjee, and U. Biswas, "An approach towards development of a new cloudlet allocation policy with dynamic time quantum," *Automatic Control and Computer Sciences*, vol. 52, no. 3, pp. 208–219, 2018, doi: 10.3103/S0146411618030033.

[23] P. Sangwan, M. Sharma, and A. Kumar, "Improved round robin scheduling in cloud computing," *Advances in Computational Sciences and Technology*, vol. 10, no. 4, pp. 639–644, 2017.

[24] S. Mittal, S. Singh, and R. Kaur, “Enhanced Round Robin Technique for Task Scheduling in Cloud Computing Environment,” *International Journal of Engineering Research and*, vol. V5, no. 10, pp. 525–529, 2016.
[25] B. Fataniya and M. Patel, “Dynamic Time Quantum Approach to Improve Round Robin Scheduling Algorithm in Cloud Environment,” *Ijsrset*, vol. 4, no. 4, pp. 963–969, 2018.
[26] A. Stephen, B. J. H. Shanthan, and D. Ravindran, “Enhanced Round Robin Algorithm for Cloud Computing,” *Int J Sci Res Comput Sci Appl Manag Stud*, vol. 7, no. 4, pp. 1–5, 2018.
[27] R. J. Matarneh, “Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes,” *American Journal of Applied Sciences*, vol. 6, no. 10, pp. 1831–1837, 2009, doi: 10.3844/ajassp.2009.1831.1837.
[28] H. Mora, S. E. Abdullahi, and S. B. Junaidu, “Modified Median Round Robin Algorithm (MMRRA),” *2017 13th International Conference on Electronics, Computer and Computation, ICECCO 2017*, vol. 2018-Janua, pp. 1–7, 2018, doi: 10.1109/ICECCO.2017.8333325.
[29] M. Runsungnoen and T. Anusas-amornkul, *Round Robin Scheduling Based on Remaining Time and Median (RR_RT&M) for Cloud Computing*, vol. 165. Springer Singapore, 2020, doi: 10.1007/978-981-15-0077-0_3.
[30] A. A. Abdelhafiz, “Al-Azhar Bulletin of Science: Section B,” vol. 32, no. 1-B, pp. 27–44, 2021.
[31] H. G. Tani and C. EL Amrani, “Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data,” *Journal of Data Mining & Digital Humanities*, vol. Special Is, 2018, doi: 10.46298/jdmdh.3104.
[32] K. ElDahshan, A. Abd, and N. Ghazy, “Achieving Stability in the Round Robin Algorithm,” *International Journal of Computer Applications*, vol. 172, no. 6, pp. 15–20, 2017, doi: 10.5120/ijca2017915161.