RESEARCH ARTICLE

An enhanced round robin using dynamic time quantum for real-time asymmetric burst length processes in cloud computing environment

Most. Fatematuz Zohora[1¤☯]*, **Fahiba Farhin**[2☯], **M. Shamim Kaiser**[3☯]

1 Computer Science and Engineering, Bangladesh Army International University of Science and Technology, Cumilla, Bangladesh, **2** Computer Science and Engineering, International University of Business Agriculture and Technology, Dhaka, Bangladesh, **3** Institute of Information Technology, Jahangirnagar University, Dhaka, Bangladesh

¤ Current address: Department of CSE, Bangladesh Army International University of Science and Technology, Cumilla, Bangladesh
☯ All these authors are contributed equally to this work.
* zohora.cse@baiust.ac.bd

Abstract

Cloud computing is a popular, flexible, scalable, and cost-effective technology in the modern world that provides on-demand services dynamically. The dynamic execution of user requests and resource-sharing facilities require proper task scheduling among the available virtual machines, which is a significant issue and plays a crucial role in developing an optimal cloud computing environment. Round Robin is a prevalent scheduling algorithm for fair distribution of resources with a balanced contribution in minimized response time and turnaround time. This paper introduced a new enhanced round-robin approach for task scheduling in cloud computing systems. The proposed algorithm generates and keeps updating a dynamic quantum time for process execution, considering the available number of process in the system and their burst length. Since our method dynamically runs processes, it is appropriate for a real-time environment like cloud computing. The notable part of this approach is the capability of scheduling tasks with asymmetric distribution of burst time, avoiding the convoy effect. The experimental result indicates that the proposed algorithm has outperformed the existing improved round-robin task scheduling approaches in terms of minimized average waiting time, average turnaround time, and number of context switches. Comparing the method against five other enhanced round robin approaches, it reduced average waiting times by 15.77% and context switching by 20.68% on average. After executing the experiment and comparative study, it can be concluded that the proposed enhanced round-robin scheduling algorithm is optimal, acceptable, and relatively better suited for cloud computing environments.

OPEN ACCESS

Citation: Zohora M.F, Farhin F, Kaiser MS (2024) An enhanced round robin using dynamic time quantum for real-time asymmetric burst length processes in cloud computing environment. PLoS ONE 19(8): e0304517. https://doi.org/10.1371/journal.pone.0304517

Editor: Qaisar Shaheen, The Islamia University of Bahawalpur Pakistan, PAKISTAN

Received: February 21, 2024

Accepted: May 14, 2024

Published: August 15, 2024

Data Availability Statement: All relevant data are within the manuscript.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Cloud Computing (CC) has become a popular technology in the modern world due to its cost savings and the scalable, accessible, and flexible services it provides for businesses and individuals [1]. This technology is aimed at reducing operational costs by providing on-demand resources such as data storage, physical and virtual servers, applications, development tools, and network capabilities via the internet [2, 3]. These resources are hosted and maintained by a remote Cloud Services Provider (CSP) that controls the user access to each resource using specific scheduling algorithms [4]. The number of resources determines the degree of simultaneous execution by a CSP. Several task scheduling algorithms such as First Come First Serve (FCFS), Shortest Job Fast (SJF), Longest Job First (LJF), Feedback Based Task Scheduling (FBTS), Priority-Based (PB), and Round Robin (RR) are incorporated to fairly and efficiently distribute these resources against each user requests [5–9]. The objective of these scheduling algorithms is to maximize resource utilization, enhance the quality of service, minimize waiting times, turnaround times, and response times, reduce costs, manage complex tasks, and ensure adaptability in a dynamic CC environment [10]. To meet this objective, researchers have proposed various scheduling algorithms considering the system's requirements.

RR is a popular scheduling approach with a fixed amount of CPU slot called Quantum Time (QT) for each process in the ready queue to execute them fairly without leaving any process waiting infinitely [11, 12]. Fig 1 displays the CC environment that uses the RR

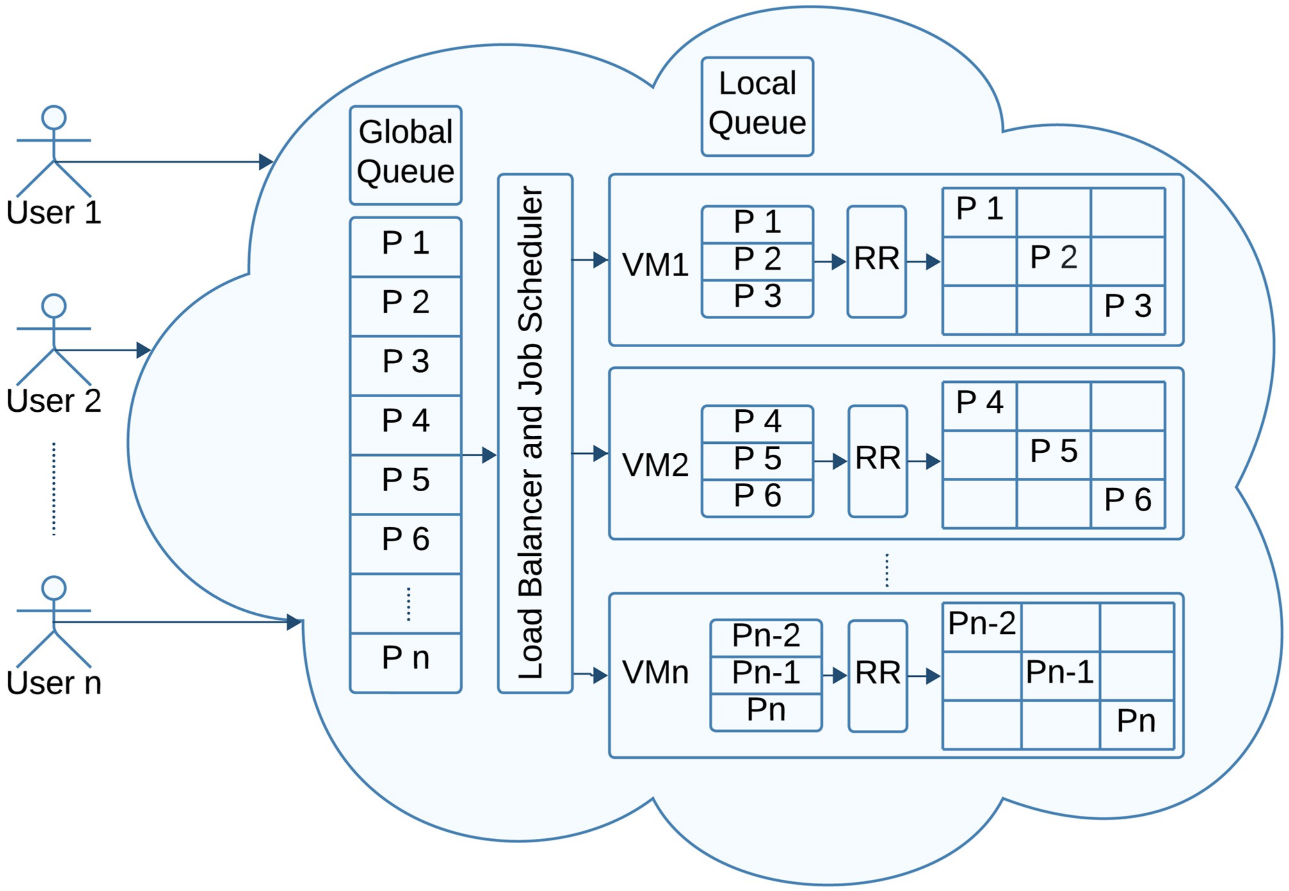

Fig 1. Round robin scheduling in cloud computing environment.

https://doi.org/10.1371/journal.pone.0304517.g001 scheduling approach to execute user tasks simultaneously. The remote user sends workload requests like data processing, computation, or any other operation to execute in the cloud environment. A job scheduler manages the allocation of resources and schedules the execution of user requests from the global queue in the distributed infrastructure. Each CSP has several virtual machines (VMs), virtualized instances of a physical computer capable of running applications and operating systems. A load balancer is employed to circulate traffic across multiple VMs equally to optimize resource utilization and ensure that no single server is overwhelmed with too much workload. VM executes user requests through multiprogramming to minimize the total execution time and incorporates different scheduling approaches. Understanding the dynamic nature of virtual machines (VMs) within CC environments is crucial for efficient task scheduling, ensuring optimal resource allocation and system performance [13]. Fig 1 shows that the VMs are using the RR algorithm to complete the tasks in a cyclic order.

The main challenge of RR is the selection of the optimal QT [14]. A small QT results in a higher Context Switching (CS) leading to additional overhead and reduced CPU efficiency whereas a large QT may increase the Average Waiting Time (AWT) of the system. The rule of thumb for selecting the optimal QT is 80 percent of the process bursts should be shorter than the QT [15]. But processes in CC arrive in the system at different times with variable length Burst Time (BT). As a result, selecting a fixed QT using this rule at the beginning may become irrelevant later. To address this issue our paper proposed a dynamic time quantum based RR approach that can manage processes with variable length BT and enhance the overall system performance reducing the AWT and unnecessary CS in the CC environment. The suggested approach improved and concentrated on a number of crucial aspects of scheduling techniques, including:

- Our technique dynamically executes processes, which makes the approach suitable for a real-time setting such as CC
- Automatically tear down the convoy effect caused by processes with relatively larger burst length and adjust the optimal QT for the existing processes in the system
- Provides a relatively small AWT, ATT, and an even distribution of context switches throughout the execution of processes

The complete proposed algorithm has been described in the methodology section with an appropriate flowchart. We have tested the proposed method on three distinct datasets: process BT in ascending order, descending order, and random order considering 20 processes for each test case. Our dataset has skewness in the data with nonuniform process BT which is a common scenario in CC. The algorithm determines QT each time a new process arrives in or leaves the system ensuring that the QT calculation considers the remaining process only. The objective of this work is to minimize the AWT and number of CS which are the crucial performance factor in the scheduling. Comparing the proposed method with some other existing improved RR algorithms it is clear that our algorithm is capable of meeting the objective for nonuniform and realistic data.

The rest of the paper is structured as follows: In section two we discussed some improved RR approaches in recent years including their outcomes and limitations. Section three describes the methodology of the proposed Enhanced Round Robin with Dynamic Time Quantum (ERRDTQ) approach with algorithm and flowchart. This section also includes the dataset we considered to evaluate our work. Experimental results and necessary discussion with appropriate tables and charts have been demonstrated in section four. Finally, the conclusion and future direction of this research is presented in section five.

Related work

Omotehinwa et al. presented a dynamic CPU scheduling algorithm (SIDRR) that uses a numeric outlier detection technique and geometric mean to determine an optimal time quantum for processes with asymmetrically distributed burst times. They implemented and tested the proposed algorithm alongside existing improved variants of Round-Robin (RR) scheduling algorithms [16–20]. For that, they feed the processes used in each paper as input into the programs written in C programming language for each of the selected algorithms to validate the correct implementation. Experimental analysis has been done based on the arrival time of processes, considering zero arrival time and non-zero arrival time categories, and different orders of burst time of processes (ascending, descending, and random order). Then they compared the performance of the proposed algorithm with the selected improved variants of RR in terms of average waiting time, average turnaround time, and number of context switches. The proposed algorithm outperformed the other variants regarding average waiting time and average turnaround time, except for EDRR, which performed better in context switching [21].

Alhaidari et al. proposed a novel technique focusing on the traditional RR algorithm disadvantages. The proposed model optimizes the functionality of the traditional RR algorithm for scheduling tasks in the cloud computing environment by optimizing the performance metrics by decreasing the average waiting time, average turnaround time, and average response time. The proposed technique is called the dynamic round-robin heuristic algorithm (DRRHA) which considers the mean of time quantum and remaining burst time of tasks. It focuses on addressing the time quantum issue by computing the average for all the tasks in the ready queue, which is organized based on the shortest job first (SJF) approach. They dynamically adjust the time quantum by dividing the calculated average by the current process's BT or remaining BT which is repeated for each task and each iteration. Additionally, it is crucial to employ the principle of checking the remaining burst time of the tasks. If the remaining burst time is less than or equal to the current task quantum, the task execution is finalized and subsequently removed from the ready queue. Otherwise, the task is stored at the end of the ready queue to be executed in the subsequent iteration. However, their algorithm calculates relatively lower QT for the larger BT in the queue creating unnecessary context switches. Numerous experiments were conducted using the CloudSim Plus tool to assess the DRRHA and then compared with related proposed algorithms [17, 22–25]. They found it outperforms other studied algorithms in terms of performance metrics like average waiting time, average turnaround time and average response time [26].

Sakshi et al. propose the Median-Average Round Robin (MARR) scheduling algorithm, which dynamically adjusts the time quantum to improve the performance compared to static Round Robin scheduling and other existing dynamic scheduling techniques. The authors compare the proposed MARR algorithm with four other scheduling algorithms to demonstrate its effectiveness [27–32]. The paper highlights the impact of the quantum decision on the scheduling of processes and the system's performance, emphasizing the need for an improved algorithm. The MARR algorithm incorporates meta-heuristic optimization strategies, making it a better-performing algorithm in scheduling techniques. The algorithm that has been proposed, along with all the performance metrics, offers the most optimal outcomes for Average Turnaround Time (ATT) and Average Waiting Time (AWT). However, the context switches remain relatively constant across all dynamic scheduling algorithms that have been considered [33].

In recent times, more research has been done based on the optimization of round-robin algorithms in cloud computing environments [34–37]. Dipto et al. proposed a new round-robin task scheduling approach named NRRTSA that enhances the performance of the allocation of resources from the central remote server in a cloud computing environment. As well as the implemented NRRTSA has also determined an efficient time quantum for each round during scheduling. They determine the time quantum dynamically based on the differences among the three maximum burst times of tasks in the ready queue for each round. It utilizes an additive manner among the differences and the burst times of the processes while determining the time quantum. By reducing average turn-around time, diminishing average waiting time, and minimizing the number of context switching, it outperforms other existing round-robin task scheduling approaches for the cloud computing environment [38].

Another study has been conducted by Nermeen et al. proposing the utilization of the ameliorated round-robin algorithm (ARRA) as a means of task scheduling in cloud computing. This algorithm has demonstrated that an ideal time quantum of (0.75 times the average) should be allocated to the tasks, with an increasing order of priority. The algorithm was then simulated and compared against other algorithms such as RR, Improved RR, Enhanced RR, ARR, and Enhanced RR (RAST ERR) [34, 39–41]. The experimental results have indicated that the ARRA algorithm has significantly reduced the Average Waiting Time (AWT) by 3.8-38.20% and the Average Turnaround Time (ATT) by 2.28-38.19% in comparison to the other algorithms [42].

Task scheduling and allocation algorithms have been thoroughly studied in recent years in cloud computing research, taking into account both single and multi-objective optimization viewpoints. In order to improve system efficiency, single-objective optimization seeks to minimize metrics like makespan (response time). Numerous studies have used novel methodologies, such as discrete Particle Swarm Optimization (PSO), hybrid Genetic Algorithm (GA), and Simulated Annealing (SA) techniques [43, 44]. In contrast, multi-objective optimization involves simultaneous consideration of multiple criteria, such as makespan minimization and monetary cost reduction [45–47]. Researchers have delved into the intricacies of trade-offs inherent in workflow scheduling to address these diverse objectives.

Although the goal of this study is to improve RR scheduling for asymmetric burst length processes in real-time in cloud computing environments, it is important to recognize the wider field of scheduling algorithms and their goals. This work attempts to integrate traditional scheduling algorithms into the cloud computing setting, in contrast to many previous research that mostly concentrate on cloud-specific solutions. Therefore, this study has not explored the single-point and multi-point optimization perspectives. However, understanding the multidimensional nature of scheduling algorithms and their implications for broader goals is critical for further study in this area.

A summary of the highlighted work done in this field is given in Table 1.

Methodology

The proposed approach in this research works on minimizing the ATT and AWT using dynamic QT which is updated each time a new process arrives in the system or a process leaves the system terminating its execution. The QT is determined considering the available burst time of all the processes in the ready queue. The QT will be updated in a manner so that 80 percent of the ready processes can complete their execution in a single turn. To facilitate the small processes earlier for maximizing the overall performance the remaining burst time of a running process will also be tracked in several checkpoints and the scheduler will decide context switching based on the remaining BT and current QT.

Definitions

This subsection presents the abbreviation used in the algorithm. Let,

$P_i = i^{th}$ number process,

$AT_i =$ Arrival Time of i^{th} number process,

Table 1. Summary of related studies.

Ref	Contributions	Time Quantum	Dataset Consideration	Limitations
[16]	Proposed an improved RR scheduling with minimizes AWT and ATT using arithmatic mean of the processes BT	QT = BT or Mean	5 processes	BT of the processes are considered as symmetric statistical distribution patterns.
[18]	This paper presents a variant of RR scheduling algorithm using dynamic QT to improve the performances	QT = Mean	5 processes in 3 different order	Processes are assumed to have BT within a specific range
[20]	Developed an efficient RR using the thumb rule of selecting QT more than 80% Processes' BT	QT = 0.8 * Maximum_BT	5 processes	Outlier BT is not considered while executing
[21]	Improved the AWT and ATT of scheduling with burst times that are asymmetrically distributed using Interquartile Range(IQR) method for outlier detection	QT = BT or Third Quartile or Arithmatic Mean or (Arrithmatic Mean + First Quartile/2)	20 processes in 3 different orders	The IQR method may fail to detect outliers in small datasets due to limited data and overidentify outliers in large datasets due to natural variability.
[23]	Proposed a variation of the RR technique that might be applied to situations where the processes' initial BT are unknown. At run time, the time quantum must be adjusted for this	QT = random number or (QT*2) or (QT/2)	5 processes	A process with a short burst time could enter in the middle of execution when the quantum is increased, which could cause the algorithm to suffer because the new process would have to wait longer than the original RR.
[26]	This paper dynamically adjusts the QT in the RR algorithm based on the mean and remaining burst time of tasks, improving task execution continuity and scheduling efficiency	QT = (Mean/2) + ((Mean/2)/BT)	5 processes in 3 different order BT range from 10 to 90	The datasets under consideration are uniformly distributed and homogeneous. For smaller processes, this approach produces significantly bigger QT, and for larger processes, smaller QT, which is irrelevant.
[32]	Developed an improved RR model using dynamic QT to gain better average response time and AWT than the classical RR	QT = (median * highest BT)	6 processes	The median rule is unable to manage processes with too short BT or too large BT.
[33]	Used a median-average based approach to set the QT dynamically for effective scheduling	QT = (Median + Mean) / 2	8 processes in 3 different order	Convoy effect results from an inability to identify the skewnesses in the BT of the processes.
[38]	The suggested method improved performance using the three highest BT of the processes in the ready queue to dynamically determine the time quantum for each round	QT = (Highest BT—1) or Highest BT	4 datasets each having 5 distinct processes	Performance degradation for processes arrived with BT in descending order.
[42]	Designed a dynamic QT to enhance overall scheduling performance based on the arithmetic mean of the process length.	QT = (3/4 * Mean)	8 processes in 3 different order	Unable to effectively manage the diversity in process BT.

https://doi.org/10.1371/journal.pone.0304517.t001

BT_i = Burst Time of i^{th} number process,
RQ = Ready Queue,
SRQ = Sorted Ready Queue,
QT = Quantum Time,
RBT = Remaining Burst Time of running process P_i,
FBT = Burst Time of the First process in the Sorted Ready Queue.

Proposed algorithm

In our proposed algorithm [Algorithm 1], Enhanced Round Robin with Dynamic Time Quantum (ERRDTQ), allows processes to join the system in real-time and add them to the ready queue, which is ordered in ascending order by process burst time. A new time quantum is also computed with the 80th percentile formula. The approach functions preemptively, applying the computed time quantum to the first process in the sorted ready queue until a new process joins the queue. The amount of burst time that the running process has left determines whether to interrupt it. The process can conclude its execution if the burst time left is less than one-third of the current time quantum. On the other hand, if this threshold is exceeded by the remaining burst duration, the algorithm looks for an alternative process in the ready queue with a burst time less than one-third of the current time quantum and assigns CPU to that process preempting the running process. The preempted process is then reintegrated into the ready queue with its updated burst time.

In this algorithm, the arrival or termination of a process works as a checkpoint for the scheduler. The design incorporates maximum 2n numbers of time quantum corresponding to n enqueue and dequeue operation, ensuring that processes with the shortest burst times are executed first, thereby minimizing average waiting time. Simultaneously, the algorithm strategically preempts running processes, taking into account their remaining burst times, thereby contributing to a balanced approach to context switching. Fig 2 depicted the detailed flow chart of our proposed algorithm.

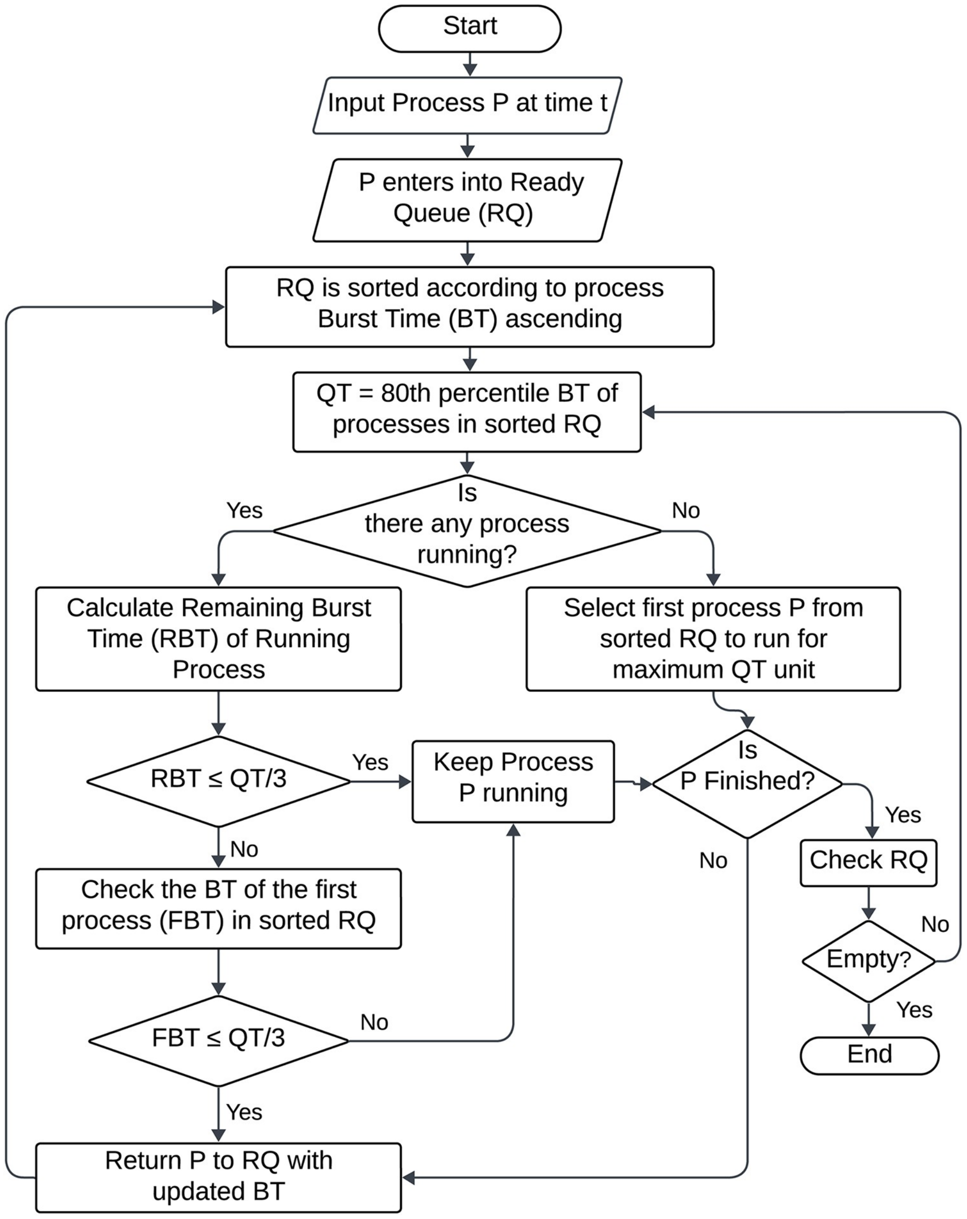
Start
Input Process P at time t
P enters into Ready Queue (RQ)
RQ is sorted according to process Burst Time (BT) ascending
QT = 80th percentile BT of processes in sorted RQ
Is there any process running?
Yes
No
Calculate Remaining Burst Time (RBT) of Running Process
Select first process P from sorted RQ to run for maximum QT unit
RBT ≤ QT/3
Yes
Keep Process P running
Is P Finished?
Yes
No
Check RQ
Empty?
No
Yes
End
No
Check the BT of the first process (FBT) in sorted RQ
FBT ≤ QT/3
No
Yes
Return P to RQ with updated BT

Fig 2. Flow chart of ERRDTQ.

https://doi.org/10.1371/journal.pone.0304517.g002

Algorithm 1 ERRDTQ

```
1: Initialize new Process $P_i$ at $AT_i$ with $BT_i$
2: Process $P_i$ enters into the RQ
3: $SRQ \leftarrow Sort(RQ)$          ▷ according to the BT in Ascending order
4: $N \leftarrow Length(SRQ)$
5: $(QT) \leftarrow SRQ[0.8 * N]$
6: if CPU is empty then
7:   Select first process P from SRQ to execute for maximum QT unit
8:   Go to step 24
9: else
10:   Calculate RBT of running Process P
11:   if RBT $\leq$ QT/3 then
12:     Keep executing P for QT unit
13:     Go to step 24
14:   else
15:     Check the FBT in the SRQ
16:     if FBT $\leq$ QT/3 then
17:       Return P to RQ with updated BT
18:     else
19:       Keep executing P for QT unit
20:       Go to step 24
21:     end if
22:   end if
23: end if
24: if P is finished then
25:   Check SRQ
26:   if SRQ is empty then
27:     End
28:   else
29:     Go to step 5
30:   end if
31: else
32:   Return P to RQ with updated BT
33: end if
```

Quantum Time (QT) calculation

The 80th percentile value displayed in [Algorithm 2] is used by the suggested ERRDTQ technique to ascertain the optimum time quantum for task scheduling. Let's look at an example that will assist in grasping this method better. Assume the following 12 values make up our dataset: [12, 45, 67, 23, 89, 34, 56, 78, 90, 10, 350, 101]. This dataset can be sorted to provide the following results: [10, 12, 23, 34, 45, 56, 67, 78, 89, 90, 101, 350]. We multiply 0.8 by 12 to find the index for the 80th percentile, and the result is 9.6. We obtain 10 by rounding this to the closest whole number. Consequently, our desired Quantum Time is represented by the 10th number of the sorted dataset, which in this case is 90.

Algorithm 2 Quantum Time Calculation

```
1: $n \leftarrow$ number of processes in SRQ
2: $index \leftarrow 0.8 * n$
3: if index is an integer then
4:   go to step 8
5: else
6:   $index \leftarrow$ nearest whole number
7: end if
8: $QT \leftarrow SRQ[index]$
```

Table 2. Processes with burst time in ascending order.

Process ID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Arrival Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Burst Time	10	15	34	37	37	44	46	51	52	55	68	71	72	74	77	79	88	91	101	350

https://doi.org/10.1371/journal.pone.0304517.t002

Table 3. Processes with burst time in descending order.

Process ID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Arrival Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Burst Time	350	101	91	88	79	77	74	72	71	68	55	52	51	46	44	37	37	34	15	10

https://doi.org/10.1371/journal.pone.0304517.t003

Table 4. Processes with burst time in random order.

Process ID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Arrival Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Burst Time	51	77	44	10	79	34	88	68	72	74	15	55	91	37	71	101	350	52	37	46

https://doi.org/10.1371/journal.pone.0304517.t004

Experimental data

The processes' arrival times convey the basis of the experimental analysis. The processes' burst times in ascending [Table 2], descending [Table 3], and random [Table 4] order were considered. We evaluated twenty processes with different arrival and burst times. The skewness of the data was taken into consideration while generating the process's burst time. The analysis was carried out under the assumptions of a single processor environment, considering burst times before execution, and non-consequential sorting time.

Results and discussion

This section demonstrates the optimal process execution by the proposed ERRDTQ approach with different datasets, and evaluates and compares the experiment result with five existing improved round-robin algorithms. The proposed method has shown comparatively outstanding performance for any kind of dataset to minimize the waiting time of the processes, balance the context switching and give a response to a waiting process in a productive manner.

Performance metrics for CPU scheduling algorithm

Numerous performance metrics are used in CPU scheduling to evaluate the efficiency of scheduling algorithms. Context switching, average waiting time, and average turnaround time are a few of the important performance indicators. To measure our proposed scheduling algorithms' effectiveness and ensure that processes are executed promptly and efficiently, these indicators are employed in this experiment.

Context switching. The technique of switching the CPU from one process to another while ensuring that the previous process's information is preserved and can be resumed later is known as context switching. In the multi-programming environment, context switching is initiated to ensure the maximum throughput of the processor. The OS can invoke context switching if a running process is waiting for an I/O or synchronization action to complete, an interrupt occurs, a transition between the user mode and kernel mode is required or a process's time quantum expires. However, context switching comes at a cost, for example, Performance overhead, Cache and Translation Lookaside Buffer (TLB) flushes, loss of energy, confusion about priorities, and even a decline in cognitive function. While designing a scheduling algorithm it is preferred to minimize the frequency of context switching as well as ensure a balanced response to the available processes.

Average Turnaround Time (ATT). The whole amount of time a process spends in the system to finish its job, including any waiting in the ready queue, is known as turnaround time. The calculation involves determining the difference between each process's arrival and termination times, then averaging these numbers. Smaller ATT indicates that processes are executing quickly.

$$ATT = \frac{\sum_{i=1}^{n}(Completion_i - Arrival_i)}{n}$$

Average Waiting Time (AWT). A significant metric in CPU scheduling is the AWT, indicating the amount of time a process spends in the ready queue for completing its execution. It is determined as the difference between a process's turnaround and burst time. The CPU scheduling procedure and process arrival sequence have a direct influence on the average waiting time. One of the primary objectives of scheduling algorithms is to reduce the overall waiting time.

$$AWT = \frac{\sum_{i=1}^{n}(Turnaround_i - Burst_i)}{n}$$

Experimental results

The overall process of the proposed ERRDTQ has been mathematically explained in this subsection. The determination of the effective time quantum for the proposed approach and the execution of all tasks with the determined effective time quantum have been described here. Intel Core i7 processor with 16 GB of RAM and 4GB Nvidia 920MX GPU were used in the preliminary test. The experiment was carried out using 3 types of datasets shown in [Tables 2–4] all having a nonzero arrival time and variable length burst time.

Case 1: Burst time in ascending order. Here, all the processes are assumed to arrive at the system with their burst time in an increasing manner. Our algorithm is designed to calculate the QT in a dynamic method with an enqueue or dequeue operation in the SRQ. Fig 3 depicted the Gantt chart of the process execution with determined QT at every checkpoint. At time 0, process A has arrived in the RQ with BT = 10 and is selected by the scheduler for execution. While executing A, 10 other processes B, C, D, E, F, G, H, I, J, and K have enqueued in the RQ at time 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 respectively. Based on their BT, all the incoming processes are sorted in ascending order and the algorithm determines a new QT each time a new process arrives. As the remaining BT of the running process (A) is smaller than one-third of the current QT, other processes are kept waiting in the SRQ. At time 10 process A left the system and process B is selected for execution as it has the smallest BT among all the available processes in SRQ and occupied the CPU till 25. In the meantime, we received 9 other processes L, M, N, O, P, Q, R, S, and T in the system at time 11, 12, 13, 14, 15, 16, 17, 18, 19 respectively. At time 25, we have 18 processes in the SRQ ready for the CPU. The scheduler selects a process with the minimum BT and executes it until no interruption occurs. The proposed ERRDTQ completed all 20 tasks with 19 context switches, 415.95 AWT and 488.55 ATT.

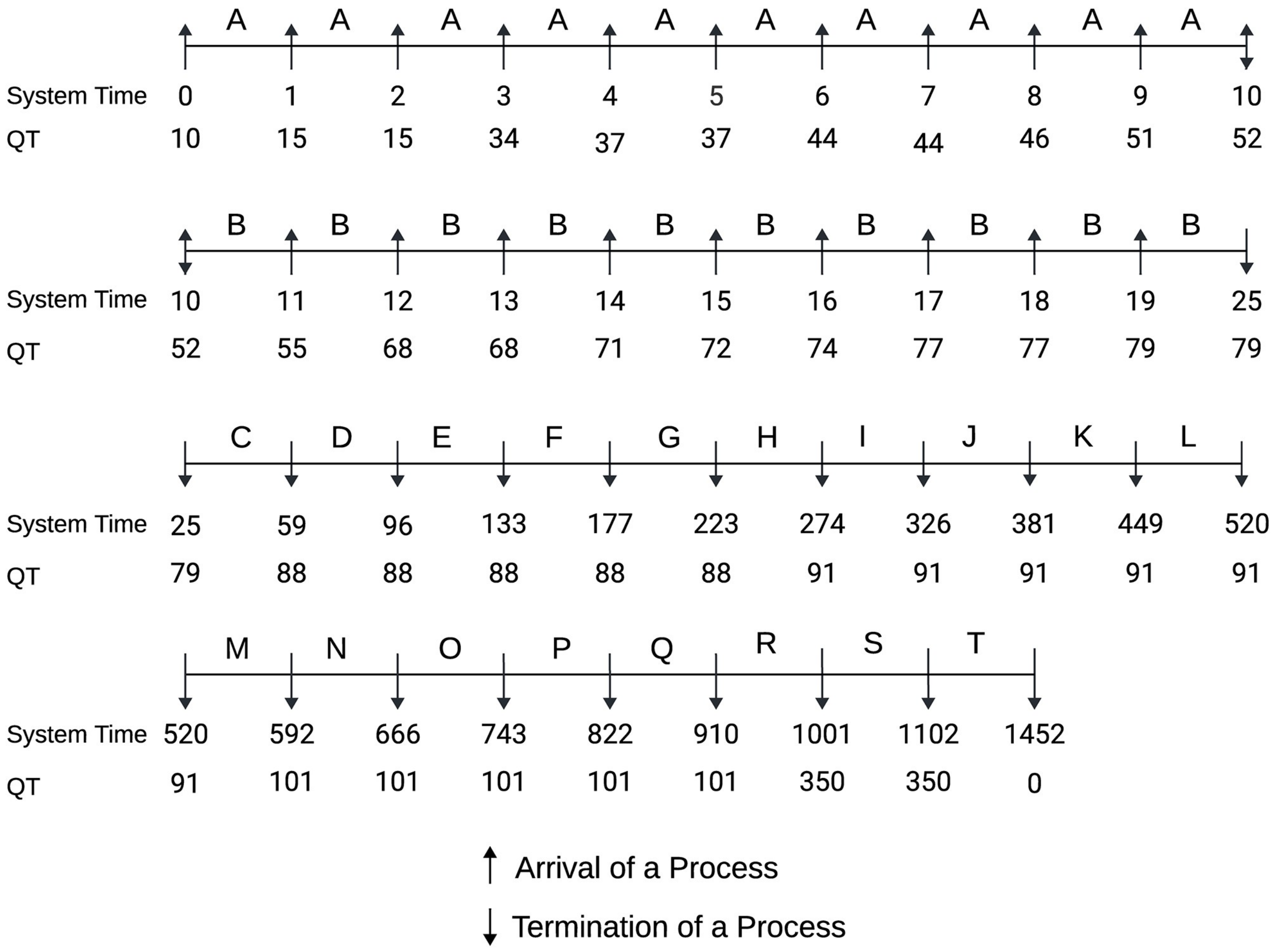

Fig 3. Gantt chart of the process with burst time in ascending order.

https://doi.org/10.1371/journal.pone.0304517.g003

Case 2: Burst time in descending order. Here, all the processes are assumed to arrive at the system with their burst time in a decreasing manner. Fig 4 depicted the Gantt chart of the process execution with determined QT at every checkpoint. At time 0, process A arrived in the RQ with BT = 350 and was selected by the scheduler for execution with QT = 350. Then, at time 1, process B arrived with BT = 101 and updated the QT as the 80th percentile BT of available processes, which is 350, and preempted process A selecting process B for execution as the BT of process B (101) is less than one-third of the current QT(117). Through this approach, we can eradicate the convoy effect that causes a small process to suffer for a comparatively large process. Process B is kept running till time 18, while we have 19 available processes (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, and S) in our RQ. The algorithm determined QT = 79 at time 18 preempted Process B and shifted CPU to Process S because Process S had BT = 15, which is less than one-third of the current QT of 27. At time 19, a new process T entered the RQ with BT = 10, updating the QT = 79. We kept process S running till its completion as the remaining BT of S is smaller than one-third of the current QT prohibiting unnecessary context switching. In this way, the scheduler kept updating the QT based on the remaining BT of ready processes and preempted the running process or let it complete its execution accordingly. The proposed ERRDTQ completed all 20 tasks with 21 context switches, 431.90 AWT and 504.50 ATT.

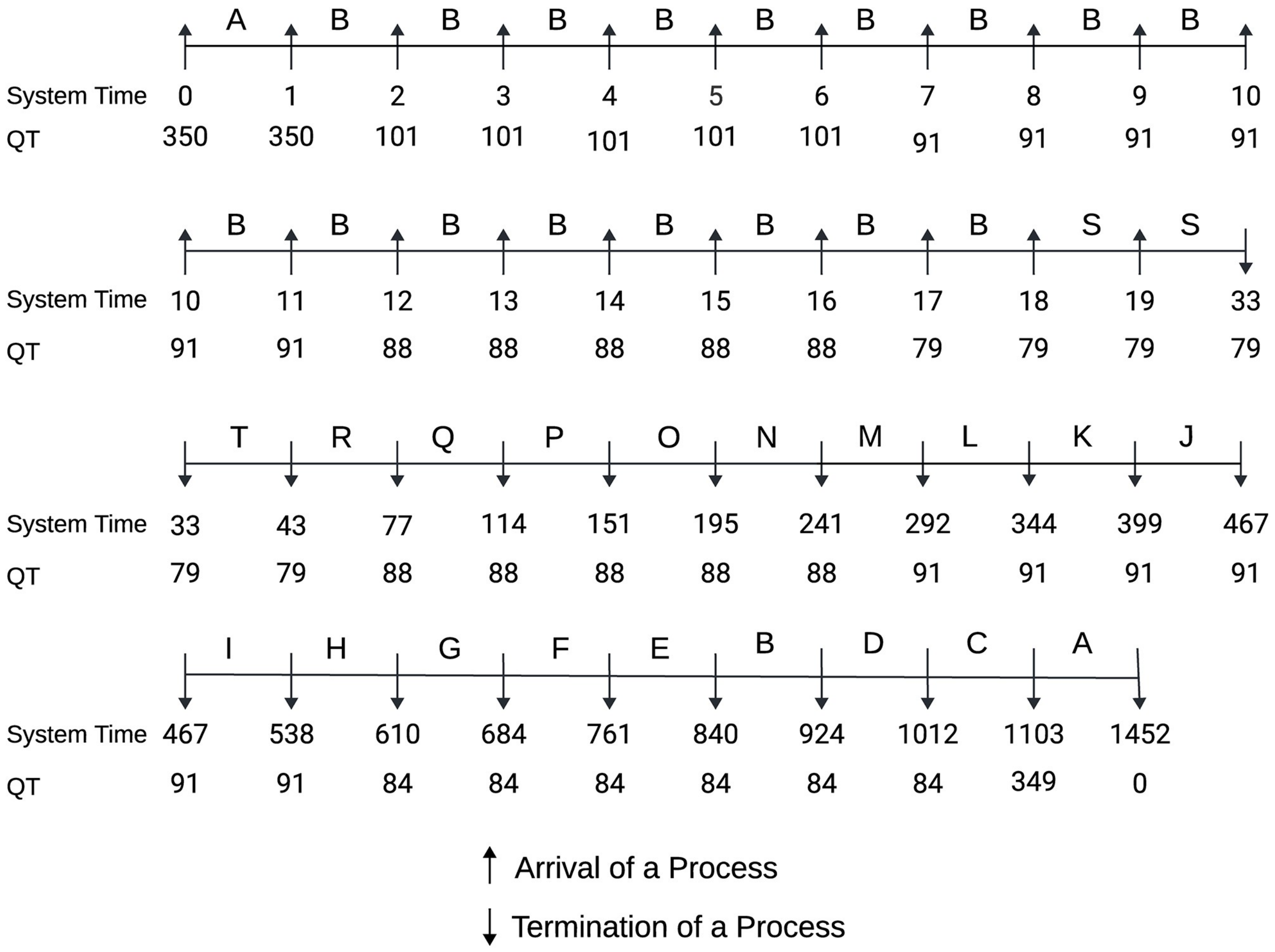
A B B B B B B B B B
System Time 0 1 2 3 4 5 6 7 8 9 10
QT 350 350 101 101 101 101 101 91 91 91 91
B B B B B B B B S S
System Time 10 11 12 13 14 15 16 17 18 19 33
QT 91 91 88 88 88 88 88 79 79 79 79
T R Q P O N M L K J
System Time 33 43 77 114 151 195 241 292 344 399 467
QT 79 79 88 88 88 88 88 91 91 91 91
I H G F E B D C A
System Time 467 538 610 684 761 840 924 1012 1103 1452
QT 91 91 84 84 84 84 84 84 349 0
Arrival of a Process
Termination of a Process

Fig 4. Gantt chart of the process with burst time in descending order.

https://doi.org/10.1371/journal.pone.0304517.g004

Case 3: Burst time in random order. Here, all the processes are assumed to arrive at the system with their burst time in a random order. Fig 5 depicted the Gantt chart of the process execution with determined QT at every checkpoint. At time 0, process A arrived in the RQ with BT = 51 and started its execution on the CPU with QT = 51. The algorithm updated QT to 77 with the arrival of process B (BT = 77) at time 1. We kept running process A because process B could not fulfil the preemption criteria. Then, at time 3, we have four processes (A, B, C, and D) in the SRQ and derived QT = 51. The scheduler preempted process A and transferred CPU to process D as it had BT = 10, which is smaller than one-third of the current QT (17). Process D terminated and left the system at time 13 when there were 13 (A, B, C, E, F, G, H, I, J, K, L, M, N) processes in the SRQ. Among them, the scheduler selected process K for execution with QT = 77. Within the completion of process K, we got all the other processes in the system and continued process execution without any further preemption until time 1452. The proposed ERRDTQ completed all 20 tasks with 19 context switches, 416.95 AWT and 489.55 ATT.

Result comparison and discussion

To evaluate the effectiveness of the proposed algorithm, we compared this with five other improved round-robin approaches. The comparison results are presented in Tables 5–7 for

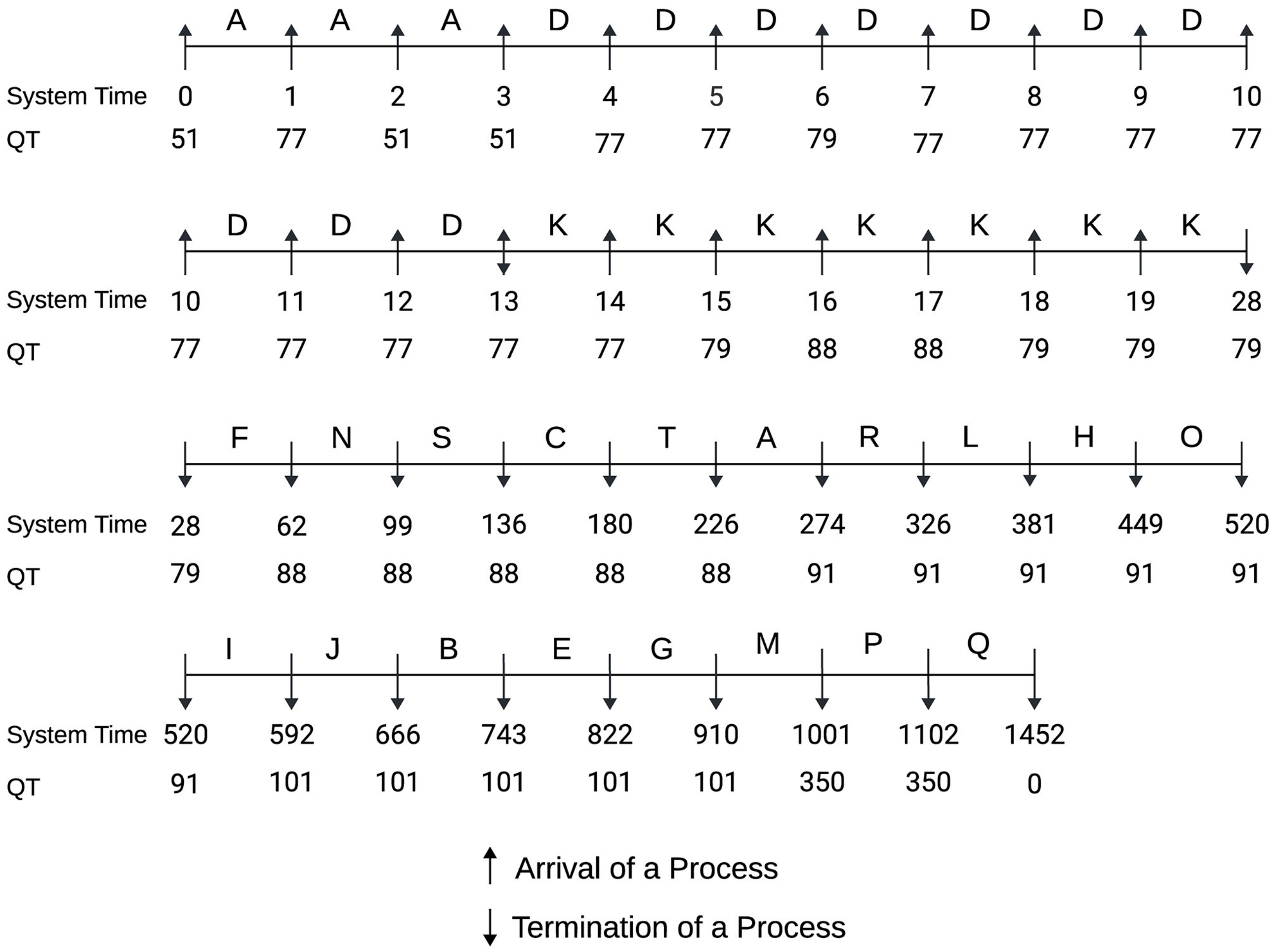

Fig 5. Gantt chart of process with burst time in random order.

https://doi.org/10.1371/journal.pone.0304517.g005

process with their burst time in ascending order (Table 2), descending order (Table 3) and random order (Table 4) respectively. Each table portrays the performance of different models considering four evaluation matrics: Time Quantum (the duration for which each process is allowed to run), AWT (the amount of time processes spend in the ready queue), ATT (the amount of time processes take to complete their tasks), and the number of context switches

Table 5. Comparative result of processes with burst time in ascending order.

Different Algorithms	SIDRR (Omotehinwa et al., 2019 [21])	DRRHA (Alhaidari et al., 2021 [26])	MARR (Sakshi et al., 2022 [33])	NRRTSA (Dipto et al., 2023 [38])	ARRA (Nermeen et al., 2023 [42])	ERRDTQ (This study)
Time Quantum	77, 350	5.5, 40.5, 39, 38, 48, 47, 133	10, 45, 74, 37	10, 67, 349	8, 33, 63	[Fig 3]
Average Waiting Time	415.95	444.45	493.95	415.95	429	415.95
Average Turnaround Time	488.55	517.05	566.55	488.55	501.6	488.55
No. of Context Switch	23	26	31	19	25	19

https://doi.org/10.1371/journal.pone.0304517.t005 (the number of times each model made processes swap in and out). The comparison shows that the proposed ERRDTQ outperforms other popular round-robin task scheduling methods by reducing AWT, ATT, and the number of context switches. In Table 5, it is depicted that the ERRDTQ algorithm performed well with the smallest AWT, ATT, and number of CS. The NRRTSA also showed the same performance for this case because shorter-length processes arrived prior in the system which automatically prevented the convoy effect (a process with a short burst time is stuck waiting for a process with a long burst time to complete).

Table 6 shows that the ERRDTQ algorithm performed better than any other approach under consideration. Here, the process with the largest BT arrived in the system first and started execution while many smaller processes kept waiting in the ready queue. As our proposed algorithm updates its QT each time a new process arrives, it could make the best decision for preempting a running process and context switch. Through this technique the ERRDTQ achieved the best AWT and ATT for processes with variable length burst time automatically preventing the outliers. For this dataset, the NRRTSA recorded the best number of CS(20), whereas our approach scored with the number of CS(21), resulting in an improved AWT.

Based on the data presented in Table 7, we can observe that the ERRDTQ algorithm outperformed the others concerning AWT, ATT, and the number of CS. In this particular scenario, the algorithm processed incoming processes with varying burst times in a random order. By selecting the most appropriate quantum time (QT), the algorithm was able to minimize both the AWT and ATT while also ensuring that unnecessary context switches were avoided.

Fig 6 presents the AWT of the six improved RR approaches including ERRDTQ. It is depicted that for each case we scored the best result. In Fig 7, the proposed algorithm's improvement rate is displayed. The algorithm resulted in a 5.05%, 35.22%, and 7.03% reduction in the average waiting time for case 1, case 2, and case 3, respectively. Moreover, it

Table 6. Comparative result of processes with burst time in descending order.

Different Algorithms	SIDRR (Omotehinwa et al., 2019 [21])	DRRHA (Alhaidari et al., 2021 [26])	MARR (Sakshi et al., 2022 [33])	NRRTSA (Dipto et al., 2023 [38])	ARRA (Nermeen et al., 2023 [42])	ERRDTQ (This study)
Time Quantum	77, 350	175.5, 29, 30, 31, 32, 26, 19, 20, 14	350, 57, 22, 13	350, 100	263, 44, 27, 17	[Fig 4]
Average Waiting Time	457.75	749.45	799.25	693.35	776.6	431.90
Average Turnaround Time	530.35	822.05	871.85	765.95	849.2	504.5
No. of Context Switch	24	32	31	20	28	21

https://doi.org/10.1371/journal.pone.0304517.t006

Table 7. Comparative result of processes with burst time in random order.

Different Algorithms	SIDRR (Omotehinwa et al., 2019 [21])	DRRHA (Alhaidari et al., 2021 [26])	MARR (Sakshi et al., 2022 [33])	NRRTSA (Dipto et al., 2023 [38])	ARRA (Nermeen et al., 2023 [42])	ERRDTQ (This study)
Time Quantum	77,350	26, 37, 38, 39,41, 47, 48, 134	51, 71, 30, 249	51,349	38,53	[Fig 5]
Average Waiting Time	422.65	450.4	522.05	422.65	438.4	416.95
Average Turnaround Time	495.25	523	594.65	495.25	511	489.55
No. of Context Switch	23	26	28	19	29	19

https://doi.org/10.1371/journal.pone.0304517.t007

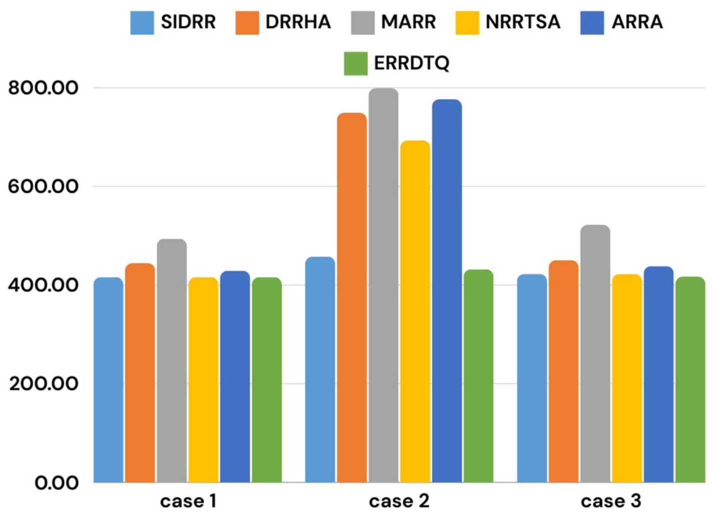

Fig 6. Comparison of AWT of different improved RR algorithms.

https://doi.org/10.1371/journal.pone.0304517.g006

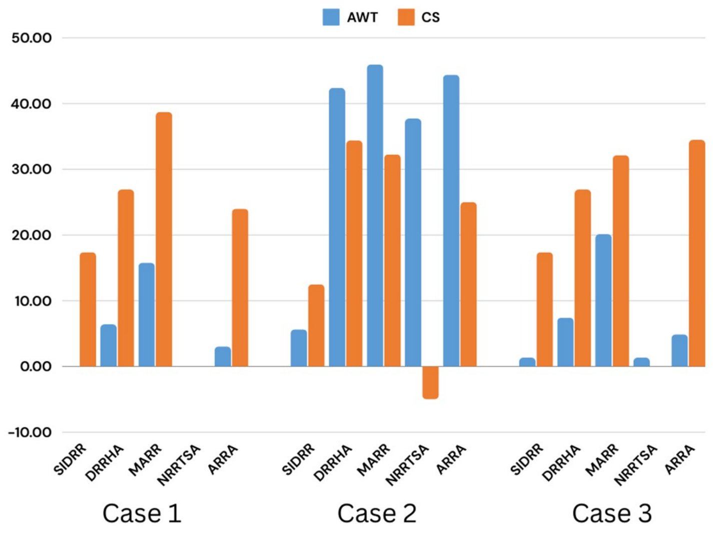

Fig 7. Improvement of ERRDTQ compared to the other related algorithms of round robin based on Average Waiting Time (AWT) and Context Switching(CS).

https://doi.org/10.1371/journal.pone.0304517.g007

recorded a 20.03%, 19.83%, and 22.19% decrease in the number of context switching for the mentioned three cases individually.

As a consequence, it is clear that the proposed approach is a cost-effective, efficient, and feasible method of task scheduling for allocating resources in a cloud computing environment. It prioritizes relatively smaller tasks to decrease the convoy effect and ensures equitable CPU time distribution while avoiding unnecessary switching overhead.

Conclusion and future direction

To maximize the utilization of resources, load balancing, reduce consumption of energy, adhere to service level agreements, and improve productivity and cost-effectiveness, optimal task scheduling is crucial in cloud computing. This paper presents a novel enhanced round-robin task scheduling method utilizing dynamic time quantum to improve the performance of task scheduling in a cloud computing environment. In addition, by automatically scheduling tasks of varying lengths and anticipating outlier burst times, the implemented ERRDTQ approach can provide minimal waiting time and a fair distribution of context switches. The results show that the suggested method performs better than the other round-robin task scheduling approaches currently in use in terms of decreasing the average waiting time, average turnaround time, and number of context switches. The study leads to the conclusion that the proposed ERRDTQ algorithm can be used in CC environment for scheduling processes with variable length BT with comparatively minimized operational cost. Our work did not incorporated the priorities of the incoming processes while planning the schedule. Our goal is to enhance the ERRDTQ methodology by taking into account the processes' priorities and balancing them with high-quality performance through the application of reinforcement learning.

Author Contributions

Conceptualization: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Data curation: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Formal analysis: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Investigation: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Methodology: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Project administration: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Resources: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Software: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Supervision: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Validation: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Visualization: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Writing – original draft: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

Writing – review & editing: Most. Fatematuz Zohora, Fahiba Farhin, M. Shamim Kaiser.

References

1. Golightly L, Chang V, Xu QA, Gao X, Liu BS. Adoption of cloud computing as innovation in the organization. International Journal of Engineering Business Management. 2022; 14:18479790221093992. https://doi.org/10.1177/18479790221093992
2. Hassan J, Shehzad D, Habib U, Aftab MU, Ahmad M, Kuleev R, et al. The rise of cloud computing: data protection, privacy, and open research challenges—a systematic literature review (SLR). Computational intelligence and neuroscience. 2022; 2022. https://doi.org/10.1155/2022/8303504 PMID: 35712069
3. Kaur J, Bahl K. Cloud Computing An on Demand Service Platform and Different Service Models. International Journal of Innovative Science, Engineering & Technology. 2018; 5(2):92–96.
4. Paul P, Aithal P, Saavedra M R, Aremu PSB, Baby P. Cloud Service Providers: An Analysis of Some Emerging Organizations and Industries. International Journal of Applied Engineering and Management Letters (IJAEML). 2020; 4(1):172–183.
5. Murad SA, Azmi ZRM, Muzahid AJM, Sarker MMH, Miah MSU, Bhuiyan MKB, et al. Priority based job scheduling technique that utilizes gaps to increase the efficiency of job distribution in cloud computing. Sustainable Computing: Informatics and Systems. 2024; 41:100942.

6. Aladwani T. Types of task scheduling algorithms in cloud computing environment. Scheduling Problems-New Applications and Trends. 2020; p. 1–12.
7. Mahmood I, M Sadeeq M, Zeebaree S, Shukur H, Jacksi K, Yasin H, et al. Task Scheduling Algorithms in Cloud Computing: A Review. Turkish Journal of Computer and Mathematics Education (TURCOMAT). 2021; 12:1041–1053. https://doi.org/10.17762/turcomat.v12i4.612
8. Banerjee P, Roy S, Sinha A, Hassan MM, Burje S, Agrawal A, et al. MTD-DHJS: makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction. IEEE Access. 2023;. https://doi.org/10.1109/ACCESS.2023.3318553
9. Murad SA, Azmi ZRM, Muzahid AJM, Bhuiyan MKB, Saib M, Rahimi N, et al. SG-PBFS: Shortest Gap-Priority Based Fair Scheduling technique for job scheduling in cloud environment. Future Generation Computer Systems. 2024; 150:232–242. https://doi.org/10.1016/j.future.2023.09.005
10. Gibet Tani H, El Amrani C. Optimization of Task Scheduling Algorithms for Cloud Computing: A Review. Lecture Notes in Networks and Systems. 2018; p. 664–672. https://doi.org/10.1007/978-3-319-74500-8_61
11. Najm NMAM. New hybrid priority scheduling algorithm based on a round Robin with dynamic time quantum. AIP Conference Proceedings. 2023; 2787(1):050018. https://doi.org/10.1063/5.0160789
12. Balharith T, Alhaidari F. Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review. In: 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS); 2019. p. 1–7.
13. Shirvani MH, Rahmani AM, Sahafi A. A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges. Journal of King Saud University-Computer and Information Sciences. 2020; 32(3):267–286. https://doi.org/10.1016/j.jksuci.2018.07.001
14. Samuel O, Oluwatosin O, Segun O. A Survey Of Variants of Round Robin Cpu Scheduling Algorithms. Fudma Journal of Sciences. 2021; 4(4):526–546. https://doi.org/10.33003/fjs-2020-0404-513
15. Peterson JL, Silberschatz A. Operating system concepts. Addison-Wesley Longman Publishing Co., Inc.; 1985.
16. Abdulrahim A, Abdullahi SE, Sahalu JB. A new improved round robin (NIRR) CPU scheduling algorithm. International Journal of Computer Applications. 2014; 90(4). https://doi.org/10.5120/15563-4277
17. Mishra MK, Rashid F. An improved round robin CPU scheduling algorithm with varying time quantum. International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol. 2014; 4.
18. Dash AR, Samantra SK, et al. An optimized round Robin CPU scheduling algorithm with dynamic time quantum. arXiv preprint arXiv:160500362. 2016;.
19. Kathuria S, Singh P, et al. A revamped mean round robin (rmrr) cpu scheduling algorithm. Int J Innov Res Comput Commun Eng. 2016; 4:6684–6691.
20. Farooq MU, Shakoor A, Siddique AB. An efficient dynamic round robin algorithm for cpu scheduling. In: 2017 International Conference on Communication, Computing and Digital Systems (C-CODE). IEEE; 2017. p. 244–248.
21. Omotehinwa T, Azeeze I, Oyekanmi E. An improved round robin cpuscheduling algorithm for asymmetrically distributed burst times. Africa Journal Management Information System. 2019; 1(4):50–68.
22. Elmougy S, Sarhan S, Joundy M. A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. Journal of Cloud computing. 2017; 6(1):1–12.
23. Muraleedharan A, Antony N, Nandakumar R. Dynamic time slice round robin scheduling algorithm with unknown burst time. Indian Journal of Science and Technology. 2016; 9(8):16. https://doi.org/10.17485/ijst/2016/v9i8/76368
24. Hemamalini M, Srinath M. Memory constrained load shared minimum execution time grid task scheduling algorithm in a heterogeneous environment. Indian Journal of Science and Technology. 2015;. https://doi.org/10.17485/ijst/2015/v8i15/71373
25. Negi S. An Improved Round Robin Approach using dynamic time quantum for improving average waiting time. International Journal of Computer Applications. 2013; 69(14). https://doi.org/10.5120/11909-8007
26. Alhaidari F, Balharith TZ. Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. Computers. 2021; 10(5):63. https://doi.org/10.3390/computers10050063
27. Shyam R, Nandal SK. Improved mean round robin with shortest job first scheduling. International Journal of advanced research in computer science and Software engineering. 2014; 4(7):170–179.
28. Banerjee P, Banerjee P, Dhal SS. Comparative performance analysis of average max Round Robin scheduling algorithm (AMRR) using dynamic time quantum with Round Robin scheduling algorithm using static time quantum. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2012; 1(3):56–62.

29. Shukla A, Simmhan Y. Model-driven scheduling for distributed stream processing systems. Journal of Parallel and Distributed Computing. 2018; 117:98–114. https://doi.org/10.1016/j.jpdc.2018.02.003

30. Lautner D, Hua X, DeBates S, Song M, Ren S. Power efficient scheduling algorithms for real-time tasks on multi-mode microcontrollers. Procedia computer science. 2018; 130:557–566. https://doi.org/10.1016/j.procs.2018.04.099

31. Noon A, Kalakech A, Kadry S. A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average. arXiv preprint arXiv:11115348. 2011;.

32. Mora H, Abdullahi SE, Junaidu SB. Modified median round robin algorithm (MMRRA). In: 2017 13th International Conference on Electronics, Computer and Computation (ICECCO). IEEE; 2017. p. 1–7.

33. Sharma C, Sharma S, Kautish S, Alsallami SA, Khalil E, Mohamed AW, et al. A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time. Alexandria Engineering Journal. 2022; 61(12):10527–10538. https://doi.org/10.1016/j.aej.2022.04.006

34. Fataniya B, Patel M. Dynamic time quantum approach to improve round robin scheduling algorithm in cloud environment. IJSRSET. 2018; 4(4):963–969.

35. Mohapatra S, Mohanty S, Rekha KS. Analysis of different variants in round robin algorithms for load balancing in cloud computing. International Journal of Computer Applications. 2013; 69(22):17–21. https://doi.org/10.5120/12103-8221

36. Pradhan P, Behera PK, Ray B. Modified round robin algorithm for resource allocation in cloud computing. Procedia Computer Science. 2016; 85:878–890. https://doi.org/10.1016/j.procs.2016.05.278

37. Tani HG, El Amrani C. Smarter round robin scheduling algorithm for cloud computing and big data. Journal of Data Mining and Digital Humanities. 2018;.

38. Biswas D, Samsuddoha M, Asif M, Ahmed MM. Optimized Round Robin Scheduling Algorithm Using Dynamic Time Quantum Approach in Cloud Computing Environment. Int J Intell Syst Appl. 2023; 15:22–34.

39. Stephen A, Shanthan BH, Ravindran D. Enhanced round Robin algorithm for cloud computing. Int J Sci Res Comput Sci Appl Manag Stud. 2018; 7(4):1–5.

40. Mittal S, Singh S, Kaur R. Enhanced round robin technique for task scheduling in cloud computing environment. Int J Eng Tech Res. 2016; 5(10):525–529.

41. Sangwan P, Sharma M, Kumar A. Improved round robin scheduling in cloud computing. Advances in Computational Sciences and Technology. 2017; 10(4):639–644.

42. Ghazy N, Abdelkader A, Zaki MS, Eldahshan KA. An ameliorated Round Robin algorithm in the cloud computing for task scheduling. Bulletin of Electrical Engineering and Informatics. 2023; 12(2):1103–1114. https://doi.org/10.11591/eei.v12i2.4524

43. Shirvani MH. A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. Engineering Applications of Artificial Intelligence. 2020; 90:103501. https://doi.org/10.1016/j.engappai.2020.103501

44. Noorian Talouki R, Hosseini Shirvani M, Motameni H. A hybrid meta-heuristic scheduler algorithm for optimization of workflow scheduling in cloud heterogeneous computing environment. Journal of Engineering, Design and Technology. 2022; 20(6):1581–1605. https://doi.org/10.1108/JEDT-11-2020-0474

45. Hosseini Shirvani M, Noorian Talouki R. Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach. Complex & Intelligent Systems. 2022; 8(2):1085–1114. https://doi.org/10.1007/s40747-021-00528-1

46. Seifhosseini S, Shirvani MH, Ramzanpoor Y. Multi-objective cost-aware bag-of-tasks scheduling optimization model for IoT applications running on heterogeneous fog environment. Computer Networks. 2024; 240:110161. https://doi.org/10.1016/j.comnet.2023.110161

47. Asghari Alaie Y, Hosseini Shirvani M, Rahmani AM. A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach. The Journal of Supercomputing. 2023; 79(2):1451–1503. https://doi.org/10.1007/s11227-022-04703-0