J Inf Process Syst, Vol.13, No.4, pp.941~950, August 2017
https://doi.org/10.3745/JIPS.01.0018

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

CPU Scheduling with a Round Robin Algorithm Based on an Effective Time Slice

Mohammad M. Tajwar*, Md. Nuruddin Pathan*, Latifa Hussaini*, and Adamu Abubakar*

Abstract
The round robin algorithm is regarded as one of the most efficient and effective CPU scheduling techniques in computing. It centres on the processing time required for a CPU to execute available jobs. Although there are other CPU scheduling algorithms based on processing time which use different criteria, the round robin algorithm has gained much popularity due to its optimal time-shared environment. The effectiveness of this algorithm depends strongly on the choice of time quantum. This paper presents a new effective round robin CPU scheduling algorithm. The effectiveness here lies in the fact that the proposed algorithm depends on a dynamically allocated time quantum in each round. Its performance is compared with both traditional and enhanced round robin algorithms, and the findings demonstrate an improved performance in terms of average waiting time, average turnaround time and context switching.

Keywords
Average Turnaround Time, Average Waiting Time, CPU Processing Time, Round Robin Algorithm, Quantum Time

1. Introduction

Many approaches are used for batch and parallel processing schemes in computing, and are aimed at improving performance [1]. Fundamental to this is process scheduling, meaning the CPU schedule for the processes or jobs available for execution [2]. This is one of the prime functions performed by the operating system. CPU scheduling is the task of selecting a process from the ready queue and allocating the CPU to it. When the CPU is idle, a waiting process is selected from the ready queue and the CPU is allocated to that process. The performance of the scheduling algorithm depends mainly on CPU utilisation, throughput, turnaround time, waiting time, response time and context switching [3]. In the round robin algorithm, a small, fixed unit of time is used, called the ‘time quantum’ or ‘time slice’ [4]. The CPU scheduler browses the ready queue, allocating the CPU to each process for a time interval up to a one-time quantum. If the CPU burst for a process exceeds the one-time quantum, the process is terminated and is returned to the ready queue.

When a new process arrives, it is added to the end of the circular queue. Compared to other existing algorithms, the RR algorithm enhances performance in the case of a time-sharing operating system. The

Manuscript received January 25, 2017; first revision April 25, 2017; accepted June 10, 2017.
Corresponding Author: Adamu Abubakar (adamu@iium.edu.my)
* Dept. of Computer Science, International Islamic University of Malaysia, Kuala Lumpur, Malaysia (mahiriium142@gmail.com, shawonpathan @hotmail.com, latifa.saee488@gmail.com)

performance of a scheduling algorithm depends upon the scheduling states of turnaround time, waiting time, response time, CPU utilisation and throughput. The turnaround time of a process is the period of time between process submission and process completion. Waiting time is the total period spent waiting in the ready queue. The time elapsed between the submission of the process and the first response received is called the response time. CPU utilisation is the percentage of time for which the CPU is busy. The number of processes completed within one unit of time is called the throughput. Context switching is the exchange of a pre-executed process from the CPU with a new process. In other words, context switching is the number of times the process switches execution. A scheduling algorithm can be optimised by reducing the response time, waiting time and turnaround time and by maximising CPU utilisation and throughput.

The uniqueness of the round robin algorithm lies in the use of a quantum time within this process, through which it has gained considerable popularity. The quantum time is of fixed size and is allocated to a job that needs to be processed. A sizeable number of research studies have defined this as a small unit of time allotted to a process that is present in the ready queue, and which can be changed depending on the resource requirements of the process [5-8]. If the process completes its execution within the required time, this implies that it will not require a further processing slot and and should be removed from the ready queue. Hence, the onward rounds of the execution of processes depend on the quantum time. Previous research studies have suggested various ways to improve the allocation of the quantum time for processes requiring execution [9], for example the use of the median method and mixed concepts [10]. These approaches, however, suffer from certain drawbacks in terms of not allowing for the modification of those processes which require a slightly greater time than the allotted time quantum cycle. This study proposes the use of a median method, which gives an improved effectiveness for the round robin algorithm by establishing a relationship which is dependent on the choice of the time quantum for processes. The time quantum for each process is dynamically allocated at each round, thus enhancing processing performance.

This paper is organised as follows. Section 2 discusses existing related work; in Section 3, we describe the proposed technique for CPU scheduling using an enhanced round robin algorithm and carry out a performance analysis. In Section 4, we analyse and compare the existing round robin algorithm and the proposed scheme in terms of context switching and computation time. Section 5 concludes this paper.

2. Related Work

As one of the easiest to implement and most starvation-free algorithms, the round robin algorithm has always been the leading choice for CPU scheduling. However, numerous researchers have proposed tweaks and adjustments to this algorithm to improve its base performance. This has led to several theories being proposed and studies conducted. The most significant of these are described below. In [11], an algorithm is proposed that allocates the CPU to all processes only in the first round, and afterwards uses the SJF algorithm to choose the next process. Another similar approach with slight differences is described in [12], where an algorithm is proposed that uses two queues called ARRIVE and REQUEST; this algorithm shows a performance improvement compared with [11]. A new algorithm called AN is proposed in [13], which is based on a new approach called the dynamic time-quantum. The idea underlying this approach that the operating system adjusts the time quantum based on the burst time of the set of waiting processes in the ready queue. The results of simulations and experiments indicate that this algorithm solves the fixed-time quantum problem and increases the performance of the RR algorithm. Similarly, in [14], an improved RR algorithm is proposed which performs optimally in timeshared systems. However, it is not suitable for soft real-time systems for a higher number of context switches, longer waiting times and higher response times. This algorithm is known as the priority based dynamic round robin algorithm (PBDRR); it calculates an intelligent time slice for each process and changes after every round of execution. The proposed scheduling algorithm is developed by taking this concept of the dynamic time quantum concept.

An optimised round robin algorithm is also proposed in [15], which consists of two phases. The first phase deals with processes which are required to be executed in order, in the same way as in the standard round robin algorithm; each process is allowed a runtime of a single time slice. During the second phase, the time quantum is doubled, and processes are executed in the order of their remaining burst times, with processes requiring shorter times being run before those with longer times. The first phase is repeated after each process is complete. Finally, in [16], an adaptive round robin algorithm is proposed which focuses on calculating an ideal time quantum. The processes are first sorted based on their burst times, with the shorter processes at the front of the ready queue, followed by a calculation of the time quantum. If the number of processes in the ready queue is even, the time quantum is equal to the average burst time of all the processes.

3. Description and Performance Analysis of the Proposed Technique

In order to create an effective CPU scheduling based on the round robin algorithm, the two important scheduling aspects of time and memory allocation are used as the control variables. The effect determines the efficiency of the proposed algorithm. However, it is worth noting here that in the round robin algorithm, context switching constitutes another essential aspect that should be taken into account. As described above, context switching involves the switching of the CPU from one process or thread to another. The proposed round robin algorithm is given in Algorithm 1.

A process or task is an executing or running instance of a program. Context switching presents a problem in the round Robin algorithm, as it causes a process to be shuffled between the ready queue and running state in the CPU several times before it is executed. The problem is more severe if the process has a very high burst time and a low static quantum time is allocated. Thus, it is essential in the round robin algorithm to reduce the number of context switches by systematically altering the time quantum. The time quantum constitutes the most appropriate section of the round robin algorithm for alteration in order to ameliorate performance.

The scheme is based on dynamic allocation of the time slice. Five processes were used to demonstrate the working of the algorithm, forming a ready queue with the five processes labelled P1, P2, P3, P4 and P5. These processes arrived at zero milliseconds (ms). The burst times (BT) of P1, P2, P3, P4, and P5 were 15, 32, 102, 48, and 29 ms, respectively. The processes were first arranged in the ascending order of burst time, resulting in the sequence P1, P5, P2, P4, P3. The time quantum was set to the mean BT for all five processes (45 ms). After executing all processes for a time quantum of 45 ms, the execution of P1, P5, and P2 was complete and these were therefore removed from the ready queue. After the first round, the remaining burst times for P3 and P4 were 3 ms and 57 ms, respectively. In the subsequent round, the new time quantum was set to the average of the remaining burst times of the processes within the ready queue, in other words to 30 ms, and the CPU was assigned to these processes using this newly calculated time quantum. After the second round was complete, process P4 had finished execution; only process P5 remained in the ready queue, with a burst time of 27 ms. Since only one process was left in the ready queue, its burst time was chosen as the time quantum and the CPU was allocated to P5. The turnaround times for P1, P2, P3, P4, and P5 were 15, 76, 226, 169, and 44 ms, respectively. The average turnaround time was 106 ms. The waiting times for P1, P2, P3, P4, and P5 were 0, 44, 124, 121, and 15 ms, respectively, and the average waiting time was 60.8 ms (see Fig. 1). The objective of introducing the dynamic average time quantum is to render the RR algorithm more efficient by reducing the number of context switches and reducing the turnaround and waiting times; this improves the performance of the round robin algorithm significantly for the processes A to G listed in Table 1, as well as the execution of the proposed algorithm, as shown in Table 2.

Algorithm 1: Proposed Round Robin

```
Begin
Define Obj fun $P(p_i),\ p = (p_1, p_2, p_3, \cdots, p_n)$
Initialise: $p_i(1, 2, \cdots, n)$ and assign it to the ready queue in ascending order (SJF). (i>0)
Set TQ = Time Quantum, BT = Burst Time
While $\left( TQ = \sum \frac{BT}{P} \in p_i \right)$ // the average of burst time of all the processes
    Do
    Execute all the processes with same TQ in the first round;
    Check $p = (p_1, p_2, p_3, \cdots, p_n)$;
      For each $p_i$
      If $(p_i(BT \le TQ))$
          Remove: $p_i$ from the queue.
      End if
      Else $(p_i(BT > TQ))$ and $p_i$ remain in ready queue
          $TQ = TQ_{new} = \left\lceil \frac{\sum_0^{n=i} p_i}{i} \right\rceil$
    End For
Repeat until all $p = (p_1, p_2, p_3, \cdots, p_n)$ are complete
End While
```

Three different time quanta, TQ1, TQ2, and TQ3, were used for the execution of processes, which clearly indicate the flow of the algorithm. Processes D to G were greater than the fixed time quantum TQ; as a result, these remained in the ready queue after the first round of execution. In the following round, a new TQ was established and used for the execution of processes D and E. Two processes (processes F and G) remained in the queue for another round and used another new TQ. The approach described above showed an increase in performance compared with the standard round robin algorithm; the performance comparison showed a significant improvement for the proposed round robin algorithm in terms of context switching, average, waiting time (WA) and average turnaround time (TA) (see Table 3).

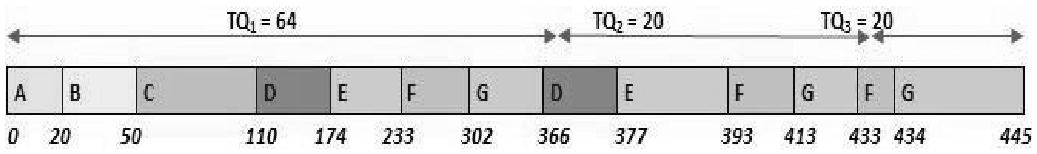


Fig. 1. Gantt chart for the processes.

Table 1. Processes and burst times

Process	Burst time (ms)
A	20
B	30
C	60
D	75
E	80
F	85
G	95

Table 2. Process execution in each round using a specific TQ

Process (Pi)	$TQ_1 = 64$	$TQ_2 = 20$	$TQ_3 = 20$
A	20	-	-
B	30	-	-
C	60	-	-
D	75	11	-
E	80	16	-
F	85	21	1
G	95	31	11
Median	N/A	18.5	6

Table 3. Process execution in each round with specific TQ

	Context switching	Average waiting time (ms)	Average turnaround time (ms)
Standard RR	16	112.20	190.70
Proposed RR	12	81.67	133.57

4. Performance Analysis

In order to examine the efficiency of the proposed algorithm, this study used performance measures and compared the performance with five other enhanced round robin algorithms, namely the improved round robin CPU scheduling algorithm with varying time quantum (IRRVQ) [12], dynamic quantum with re-adjusted round robin scheduling algorithm (DQRRR) [17], self-adjustment time quantum in round robin (SARR) [18], and modified round robin algorithm (MRR) [19]. All the algorithms were tested in terms of their burst time in ascending order, burst time in descending order and burst time in random order. The performances of the algorithms were then compared in terms of context switching, average waiting time and average turnaround time. This comparison was carried out based on the fact that only CPU-bound processes were considered in the analysis. Five independent processes were analysed for each test case. The burst times and arrival times of processes were known before execution, and the context switching times of processes were assumed to be zero. The time required to arrange the processes into ascending order was also considered to be zero. Three different cases were considered for testing (see Tables 4–10).

Table 4. Case 1: processes in ascending order

Process	Burst time (ms)
P1	40
P2	55
P3	60
P4	90
P5	102

Table 5. Performance comparison in ascending order

	RR	DQRRR	IRRVQ	SARR	MRR	This study
TQ	25	60/36/6	40/15/5/30/12	60/36/6	62/25/25	69/27/6
Context switching	16	7	14	7	8	7
Avg. waiting time	192	162.2	165	119	124.4	120.8
Avg. turnaround time	261.4	231.6	234.4	188.4	193.8	190.2

Table 6. Case 2: processes in descending order

Process	Burst time (ms)
P1	105
P2	85
P3	55
P4	43
P5	35

Table 7. Performance comparison in descending order

	RR	DQRRR	IRRVQ	SARR	MRR	This study
TQ	25	55/40/10	35/8/12/30/20	55/40/10	70/25/25	64/31/10
Context switching	15	7	14	7	7	7
Avg. waiting time	209.4	144.8	142	185.8	106.4	105.6
Avg. turnaround time	274	209.4	206.6	250.4	171.4	170.2

Table 8. Case 3: processes in random order

Process	Burst time (ms)
P1	105
P2	60
P3	120
P4	48
P5	75

Table 9. Performance comparison in random order

	RR	DQRRR	IRRVQ	SARR	MRR	This study
TQ	25	75/37/8	48/12/15/30/15	120	72/45/25	81/31/8
Context switching	17	7	14	4	8	7
Avg. waiting time	245	192.8	193.2	177.6	168.6	141.6
Avg. turnaround time	327	274.4	274.8	259.2	250.2	223.6

Table 10. Cumulative performance comparison

	RR	DQRRR	IRRVQ	SARR	MRR	This study
Context switching	56	21	42	18	23	21
Avg. waiting time	646.8	494.8	500.2	482.4	399.8	368
Avg. turnaround time	862.4	715.4	716	698	615.4	584

Tables 4–10 above summarise the test cases in which the various round robin techniques were compared with the proposed round robin algorithm. The values of average turnaround time, waiting time and context switching for the proposed algorithm were found to be superior to those of the other algorithms. All algorithms were tested based on a zero arrival time for the processes. A decrease was found in all three performance metrics used to test the algorithms. There was a marked decrease in times for the proposed algorithm, as both turnaround and waiting times decreased significantly. The number of context switches for the proposed algorithm was similar to that of SARR, but was still relatively low compared to the remaining algorithms. In view of this, it can be concluded that the proposed algorithm has superior efficiency in terms of execution time and context switching in all three cases of process execution. Compared to the traditional round robin algorithm, the proposed algorithm achieves a saving of 41% in waiting time compared to the other variants of the algorithm, thus proving its effectiveness.

The algorithm works slightly differently when the processes arrive at different times. To consider this scenario, we assume that if there are a finite number of tasks in the scheduler, with different arrival times, the first step implements the traditional round robin and takes the burst time of the first process as the time quantum. Once all the processes have run for the first cycle, we then implement our proposed algorithm by arranging the remaining burst times in ascending order. This gives rise to three possible cases. We assume that three processes (A, B, C) arrive at three different times, with burst times of 10, 8, and 6 ms. Here, the TQ is set to 10 ms for the first cycle. This is the best case scenario for the algorithm, as the burst times of the remaining processes are less than the TQ and can therefore be implemented directly in the first cycle. In the worst case, the sequence of burst times is 6, 8, and 10 ms; here, the TQ is 6 ms, which is the lowest of the burst times. To show that the proposed algorithm works in the worst case, an example is given below.

Tables 11 and 12 provide additional scenarios in which a performance comparison shows that there is a significant improvement in the proposed round robin in terms of context switching, average waiting time (WA) and average turnaround time (TA). For a scenario involving varying arrival times, the algorithm reduces the number of context switches by more than 50% compared to the traditional round robin. For each scenario, we assume that the burst times for the processes are normally distributed and can be mathematically determined using statistical models such as the Poisson distribution [20] (see Fig. 2).

Table 11. Remaining burst times for first cycle (traditional round robin) TQ

Process	Arrival time (ms)	Burst time (ms)	Remaining time (ms)
A	0	20	0
B	10	200	180
C	12	150	130
D	21	230	210
E	26	75	55

Table 12. Process execution at each round with specific TQ

Process	Arrival time (ms)	Burst time (ms)
E	55	–
C	130	–
B	130	36
D	210	66

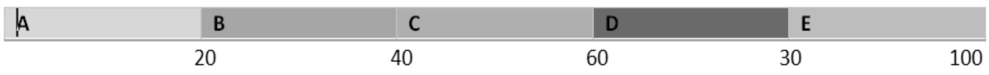


Fig. 2. Gantt chart for the processes in the first cycle.

5. Conclusions

This paper presents an enhanced round robin algorithim which aims to circumvent the weaknesses of previous studies on efficient and effective CPU scheduling. Many studies use the required processing time for a CPU execution in order to determine the optimal scheduling technique. As a consequence, these studies suggest various CPU scheduling algorithms according to a processing time which is based on different criteria. This research acknowledges the fact that the round robin algorithm enjoys more popularity than any other CPU scheduling algorithm based on its optimal time-shared environment. We show that the proposed MRR enhances the effectiveness of the original round robin algorithm and its existing techniques. To evaulate its performance, the proposed algorithm was compared to the standard round robin algorithm. The findings indicate an improved performance in terms of average waiting time, average turnaround time and context switching.

Acknowledgement

This paper was supported by a Research Initiative Grant Scheme (No. RIGS16-364-0528) from the International Islamic University of Malaysia (IIUM).

References

[1] B. Kim, C. H. Youn, Y. S. Park, Y. Lee, and W. Choi, "An adaptive workflow scheduling scheme based on an estimated data processing rate for next generation sequencing in cloud computing," *Journal of Information Processing Systems*, vol. 8, no. 4, pp. 555-566, 2012.

[2] S. M. Khorandi and M. Sharifi, "Scheduling of online compute-intensive synchronized jobs on high performance virtual clusters," *Journal of Computer and System Sciences*, vol. 85, pp. 1-17, 2017.

[3] C. Rani and M. M. Bala, "Comparison of round robin based CPU scheduling algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 9, pp. 28-33, 2015.

[4] E. Hyytia and S. Aalto, "On round-robin routing with FCFS and LCFS scheduling," *Performance Evaluation*, vol. 97, pp. 83-103, 2016.

[5] F. Jafari, A. Jantsch, and Z. Lu, "Weighted round robin configuration for worst-case delay optimization in network-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 12, pp. 3387-3400, 2016.

[6] K. Mahajan, A. Makroo, and D. Dahiya, "Round robin with server affinity: a VM load balancing algorithm for cloud based infrastructure," *Journal of Information Processing Systems*, vol. 9, no. 3, pp. 379-394, 2013.

[7] R. V. Rasmussen and M. A. Trick, "Round robin scheduling: a survey," *European Journal of Operational Research*, vol. 188, no. 3, pp. 617-636, 2008.

[8] A. Bisht, M. A. Ahad, and S. Sharma, "Enhanced round robin algorithm for process scheduling using varying quantum precision," in *Proceedings of ICRIEST AICEEMCS*, Pune, India, 2013, pp. 11-15.

[9] D. Nayak, S. K. Malla, and D. Debadarshini, "Improved round robin scheduling using dynamic time quantum," *International Journal of Computer Applications*, vol. 38, no. 5, pp. 34-38, 2012.

[10] L. Datta, "Efficient round robin scheduling algorithm with dynamic time slice," *International Journal of Education and Management Engineering*, vol. 2, pp. 10-19, 2015.

[11] R. K. Yadav, A. K. Mishra, N. Prakash, and H. Sharma, "An improved round robin scheduling algorithm for CPU scheduling," *International Journal of Computer Science and Engineering*, vol. 2, no. 4, pp. 1064-1066, 2010.

[12] M. K. 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.

[13] A. Noon, A. Kalakech, and S. Kadry, "A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average," *International Journal of Computer Science Issues*, vol. 8, no. 1, pp. 224-229, 2011.

[14] R. Mohanty, S. R. Behera, and S. C. Pradhan, "A priority based dynamic round robin with deadline (PBDRRD) scheduling algorithm for hard real time operating system," *International Journal of Advanced Research in Computer Science*, vol. 3, no. 3, pp. 186-192, 2012.

[15] C. McGuire and J. Lee, "Comparisons of improved round robin algorithms," in *Proceedings of the World Congress on Engineering and Computer Science*, San Francisco, CA, 2014, pp. 1-4.

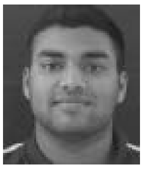
[16] M. Lee, S. Song, J. Moon, J. Kim, W. Seo, Y. Cho, and S. Ryu, "Improving GPGPU resource utilization through alternative thread block scheduling," in *Proceedings of 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Orlando, FL, 2014, pp. 260-271.

[17] H. S. Behera, R. Mohanty, and D. Nayak, "A new proposed dynamic quantum with re-adjusted round robin scheduling algorithm and its performance analysis," *International Journal of Computer Applications*, vol. 5, no. 5, pp. 10-15, 2010.

[18] R. J. Matarneh, "Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes," *American Journal of Applied Science*, vol. 6, no. 10, pp. 1831-1837, 2009.

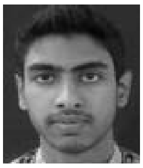
[19] N. Mittal, K. Garg, and A. Ameria, "A paper on modified round robin algorithm," *International Journal of Latest Technology in Engineering, Management & Applied Science*, vol. 4, no. 11, pp. 93-98, 2015.

[20] Y. Liang, "A simple and effective scheduling mechanism using minimized cycle round robin," in *Proceedings of 2002 IEEE International Conference on Communications*, New York, NY, 2002, pp. 2384-2388.



Mohammad M. Tajwar

He is currently a B.S. Student in the Department of Computer Science at the International Islamic University of Malaysia. His current research interests include artificial intelligence and parallel computing.

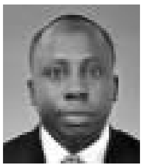


Md. Nuruddin Pathan

He is currently a B.S. Student in the Department of Computer Science at the International Islamic University of Malaysia. His research interests include machine learning and parallel computing.

Latifa Hussaini

She is currently a B.S. Student in the Department of Computer Science at the International Islamic University of Malaysia. Her current research interests include machine learning and parallel computing.



Adamu Abubakar http://orcid.org/0000-0002-9137-3974

He received B.S. and M.S. degrees in Geography and Computer Science from the School of Science at Bayero University, Kano, Nigeria, in 2004 and 2006, respectively, and a Ph.D. in Information Technology from the Department of Information and Communication Technology, International Islamic University of Malaysia. Since November 2012, he has been an Assistant Professor within the Department of Computer Science at the International Islamic University of Malaysia. His current research interests include machine learning and parallel computing.