International Journal of Scientific & Engineering Research, Volume 8, Issue 12, December-2017
ISSN 2229-5518

Improved Performance of Round Robin CPU Scheduling Algorithm Using Non-preemptive SJF

Md. Rashid Al Asif, Mehedi Raihan, Md. Zakaria Hossain, Md. Alam Hossain, Md. Abdul Momin

Abstract—The performance of multitasking operating system heavily depends on its scheduling algorithms. Round Robin is the most important part of CPU scheduling algorithm in the operating system where time quantum affects the performance. This algorithm is very useful for CPU scheduling that gives equal time quantum to all processes. This paper presents an improvement of RR CPU scheduling algorithm that reduces the average turnaround time, average waiting time and the number of context switches. The RR CPU scheduling algorithm with non-preemptive SJF method is applied for getting better performance. The proposed algorithm works as follows (i) Round Robin CPU scheduling algorithm is applied until all the processes arrive in the request queue, (ii) When all the processes are in the request queue then non-preemptive SJF is applied for remaining process execution. This proposed method increases the performance of RR CPU scheduling algorithm.

Index Terms— Burst Time, Context Switching, CPU Scheduling Algorithm, Improved Performance of Round Robin, IPRR, Non-preemptive, Operating System, Process Scheduling, Round Robin Scheduling, Shortest Job First, SJF, Turnaround Time, Waiting Time.

1 Introduction

An operating system is a program that manages computer hardware system. CPU scheduling algorithms play a vital role in the computer operating system. In operating system multi-programming concept is very important to increase CPU performance. Multitasking is one of the most important parts supported by modern operating system. Multi-programming increases CPU utilization by organizing processes. Only one process can run in the CPU at a time in a single processor system, other processes in the ready queue have to wait until the CPU becomes free to execute the next process. A fundamental operating function is scheduling. Before using almost all computer resources are scheduled. Decision of scheduling tries to reduce turnaround time, average waiting time and response time for processes and the number of context switches.

2 CPU Scheduling algorithms

There are several basic CPU scheduling algorithms, which are First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling (PS) and Round Robin (RR).

2.1 FCFS

FCFS is a simple scheduling algorithm. In this technique, processes are executed on first come first serve basis. The implementation of this method is easily managed with a FIFO queue. It allocates the processes to CPU according to their arrival in the ready queue. When the CPU is free then a process is removed from the front of ready queue and gets CPU allocation. The average waiting time of this technique is high [8].

2.2 SJF

According to the burst time, the process moves to the ready queue. The process with minimum burst time gets CPU allocation when the CPU is available. If the burst time of next two processes has the same value then FCFS rule is applied to those processes. A non-preemptive SJF does not relinquish CPU allocation until finishes it's currently running CPU burst. But preemptive SJF do the opposite [2].

2.3 PS

Every process is assigned a priority and the process is allocated according to their priority (highest to lowest). If multiple processes have the same priority then the FCFS rule is applied for the processes [8], [10].

2.4 RR

Round Robin is specially design for the time sharing and real time operating system. Each process maintains a small time unit called time quantum. In this fashion, the processes get CPU allocation for one time quantum at a time. When the process needs more time, the process runs for the full length of the time quantum and that process will be preempted and then added to tail (rare) of the queue. Hence, the RR works in preemptive fashion [2], [8], [11].

3 Scheduling criteria

Different CPU scheduling algorithms have different properties which decide selection of process using various criteria for execution by CPU. Those criterions are used for comparison and decide how one algorithm differs from the other are given below:

1. Throughput: It is the number of process completed per unit time.
2. Turnaround Time: It means to take total time of the CPU to execute a process. It can be calculated as Turnaround Time = Completion Time – Arrival Time or Burst Time + Waiting Time.
3. Waiting Time: This is how much time a process has been waiting in ready queue. Waiting Time = Turnaround Time – Burst Time.
4. CPU Utilization: CPU utilization measures how much busy the CPU.
5. Context Switch: Switching of the CPU from one process to another is Context Switching.

6. Response Time: It is the total amount of time to take CPU at first time from the time of entering.

So, the following characteristics may represent a good CPU scheduling algorithm [1], [8]:

1. Minimum context switches.
2. Maximum CPU utilization.
3. Maximum throughput.
4. Minimum turnaround time.
5. Minimum waiting time.
6. Minimum response time.

4 Literature review

Round Robin CPU scheduling algorithm is mostly used in and real-time and timesharing operating systems. The performance of this algorithm is heavily dependent on quantum time. So, the selection of quantum time is crucial. Quantum time can be selected statistically or dynamically. Several authors have been proposed various modifications of RR algorithm.

From [12], every process is allocated to CPU on a priority basis for only one quantum time. After that, remaining burst time of each process assign a priority and performs scheduling in SJF fashion. The shortest remaining burst time possess the highest priority.

In the articale [13], all the processes are arranged in ascending order according to their burst time. Then, by taking the median (as quantum time) of processes that present in the ready queue. Afterwards, time quantum is recalculated for the remaining burst time of processes.

Kishor et al [14] perform a modification of RR algorithm with zero arrival time (when t = 0 then all the processes arrive). It works like as [13] except the time quantum is calculated by taking the average of burst times in the ready queue. And, the same technique is applied for the remaining burst time of processes.

A new algorithm with zero arrival time is proposed by Hiranwal and Roy [15]. Here all the processes are arranged in ascending order according to their burst time. After that, time quantum is calculated which depends on a number of processes. If the number of processes is odd then take the burst time of mid-process as quantum time otherwise take the average of burst times as quantum time.

Another paper proposed Longest Job First Combinational Burst Time (LJF+CBT) CPU scheduling algorithm which is an improvement of Longest Job First (LJF). This algorithm arranges the processes in descending order according to their burst time and determines the average of processes as Combined Weighted Average (CWA). CWA is a criterion by which a short or long process is identified. If the burst time of a process is greater than CWA then it is recognized as a long process otherwise short process. Two consecutive shorter processes are used to create new burst time, merged into one, placed and sort in a new queue in descending order according to their burst time. Afterwards, processes are allocated to CPU in LJF fashion [16].

Mishra [7] introduced an algorithm (IRR) where each process is allocated in RR fashion. Afterwards, it checks the remaining burst time of currently running process. If the burst time is less than the time quantum then it gets the CPU allocation again otherwise moves to the end of the READY queue.

Abdulrahim et al [8] added an ARRIVE queue to a modification of IRR and consider the ceiling of the average of burst time of processes as quantum time. If a running process has a burst time more than half of the quantum time then it moved from REQUEST queue to ARRIVE queue. For the remaining burst time of processes, the time quantum is calculated again. And this iteration will continue until REQUEST and ARRIVE queue is empty.

Sain [3] proposed Dynamic time quantum Shortest Job Round Robin (Dynamic SJRR) where half of the burst time of the first process is considered as the time quantum for the first process. Then the processes are arranged in ascending order according to their F (where F = arrival time + CPU burst time). Now, for all the processes that has a burst time equal to F is allocated to CPU in FCFS manner. After that, quantum time is taken from the mean value (which is derived from burst times of all processes). Finally, new quantum is assigned to all processes and recalculate if the remaining burst time exist. And the procedure iterates unit the ready queue is empty.

Datta [17] performs a modification of DRRR algorithm [13] by adding different quantum time for each round of RR CPU scheduling. And this quantum time is calculated by considering the remaining CPU burst time, waiting time for each process. For a real-time system, it also facilitates the implementation of simple RR comparing with [13].

Authors represent this paper as a modification of RR CPU scheduling algorithm. The proposed algorithm checks all the processes are in request queue or not. If yes then it applies non-preemptive SJF for all the remaining processes including the remaining burst time of the running process. As a result, the proposed algorithm performs better by reducing average turnaround time (ATAT), average waiting time (AWT), and number of context switches (NCS).

5 Proposed algorithm

The proposed Improved Performance of Round Robin (IPRR) CPU scheduling algorithm maintains two queues namely, request queue and ready queue. At first, processes are placed in request queue according to their arrival time. For getting CPU allocation, a process is moved from request queue to ready queue. The processes start execution in RR fashion until all the processes arrive in request queue. If the burst time of a running process is greater than quantum time then the process with remaining burst time is moved from ready queue to request queue. When all the processes are available in request queue then non-preemptive SJF is applied for remaining processes. The algorithm comprises the following steps:

Step 1: Insert all the processes in request queue according to their arrival time. For CPU allocation a process is moved from request queue to ready queue.

Step 2: Round Robin CPU scheduling algorithm is applied for process execution until all the processes arrive in request queue. If the burst time of a running process is greater than time quantum then the process with remaining burst time is moved from ready queue to request queue.

Step 3: When all the processes are available/last process

arrive in request queue then non-preemptive SJF is applied for remaining processes.

Step 4: Calculate average waiting time, average turnaround time and number of context switches.

5.1 Illustrative Example of Proposed IPRR Algorithm

Table 1 is used to illustrate the RR and the proposed IPRR algorithm. All the processes are arrived in different arrival time. And a request queue is maintained for process selection. The time quantum is taken 3ms for both RR and IPRR.

Table 1

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	3	6
P4	5	1
P5	6	4

Calculation of average waiting time, average turnaround time and number of context switches for RR:

Request queue for RR: P1, P2, P3, P1, P4, P5, P3, P5

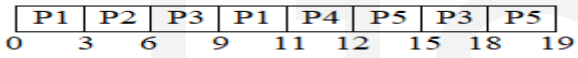

Fig. 1: Gantt chart of RR

Waiting Time:

P1: (0 - 0) + (9 - 3) = 6,
P2: (3 - 1) = 2,
P3: (6 - 3) + (15 - 9) = 9,
P4: (11 - 5) = 6,
P5: (12 - 6) + (18 - 15) = 9.

Average Waiting Time (AWT) = (6 + 2 + 9 + 6 + 9)/5 = 6.4

Turnaround Time:

P1: (5 + 6) = 11,
P2: (3 + 2) = 5,
P3: (6 + 9) = 15,
P4: (1 + 6) = 7,
P5: (4 + 9) = 13.

Average Turnaround Time (ATAT) = (11 + 5 + 15 + 7 + 13)/5 = 51/5 = 10.20

Number of context switches (NCS) = 7

Calculation of average waiting time, average turnaround time and number of context switches for IPRR:

Request queue for IPRR: P1, P2, P3, P1, P4, P5

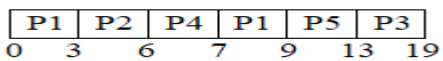

Fig. 2: Gantt chart of PIRR

Waiting time:

P1: (0 - 0) + (7 - 3) = 4,
P2: (3 - 1) = 2,
P3: (13 - 3) = 10,
P4: (6 - 5) = 1,
P5: (9 - 6) = 3.

Average Waiting Time (AWT) = (4 + 2 + 10 + 1 + 3)/5 = 20/5 = 4

Turnaround Time:

P1: (5 + 4) = 9,
P2: (3 + 2) = 5,
P3: (6 + 10) = 16,
P4: (1 + 1) = 2,
P5: (4 + 3) = 7.

Average Turnaround Time (ATAT) = (9 + 5 + 16 + 2 + 7)/5 = 39/5 = 7.80

Number of context switches (NCS) = 5

Table 2: Comparison of RR and IPRR

Grantt	Time Quantum	AWT	ATAT	NCS
RR	3	6.40	10.20	7
Proposed IPRR	3	4	7.80	5

From Table 2, the proposed IPRR algorithm shows better performance than Round Robin CPU scheduling algorithm.

5.2 Evaluation of Proposed IPRR Algorithm

The processes shown in Table 3, Table 5 and Table 7 were used to evaluate the proposed algorithm. There are some processes and their arrival time and burst time corresponding.

Table 3

Process	Arrival Time	Burst Time
P0	0	15
P1	2	13
P2	4	8
P3	5	10
P4	1	12
P5	6	26
P6	3	7

Time Quantum = 5ms

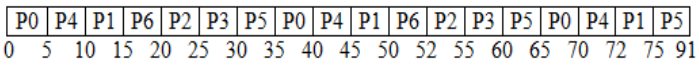

Fig. 3: Gantt chart of RR

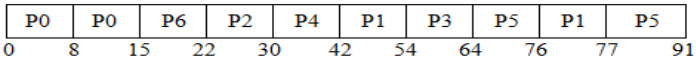

Fig. 4: Grantt chart of dynamic SJRR

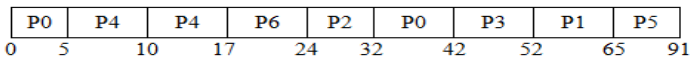

Fig. 5: Grantt chart of proposed IPRR

Table 4: Comparison of RR, Dynamic SJRR and IPRR

Grantt	Time Quantum	AWT	ATAT	NCS
RR	5	51.85	64.85	17
Dynamic SJRR	8, 12, 7	32.28	45.28	8
Proposed IPRR	5	30.14	43.14	8

Now, consider a dataset with zero arrival time.

Table 5

Process	Arrival Time	Burst Time
P1	0	23
P2	0	75
P3	0	93
P4	0	48
P5	0	2

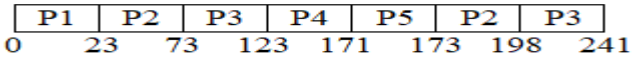

Fig. 6: Grantt chart of RR with time quantum = 50ms

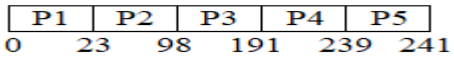

Fig. 7: Grantt chart of IRR with time quantum = 50ms

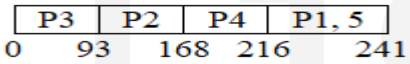

Fig. 8: Grantt chart of LJF+CBT

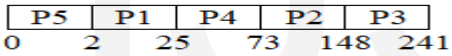

Fig. 9: Grantt chart of proposed IPRR

Table 6: Comparison of RR, IRR, LJF+CBT and IPRR

Algorithm	AWT	ATAT	NCS
RR	113	161.2	6
IRR	110.2	158.4	4
LJF+CBT	95.4	143.6	3
Proposed IPRR	49.6	97.8	4

Now, consider another dataset with non-zero arrival time:

Table 7

Process	Arrival Time	Burst Time
P1	0	28
P2	2	35
P3	6	50
P4	6	82
P5	8	110

Lipika Datta [17] proposed a dynamic time slice algorithm and comparing with DQRRR as in [13] using Table 7 which performs better in average wating time, average turnaround itme and fewer number of context switches.

The authors also inverstigate Table 7 with proposed IPRR alogorithm where 20ms taken as time quantum.

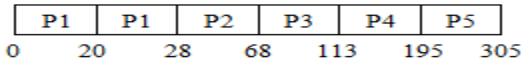

Fig. 10: Grantt chart of proposed IPRR

Table 8: Comparision of DQRR, Dynamic Time Slice and IPRR

Algorithm	AWT	ATAT	NCS
DQRRR	112.2	173.2	7
Dynamic Time Slice	94.6	152.0	7
Proposed IPRR	79.8	140.8	5

6 Result and Analysis

To determine the performance of CPU scheduling algorithm AWS, ATAT, NCS are considered as crutial factor. The proposed IPRR algorithm results better compared with the reference papers (discussed in literature review section) which are evaluated in section 5.2.

The following graphical resentation show the comparision of AWT, ATAT, and NCS of different algorithms with proposed one.

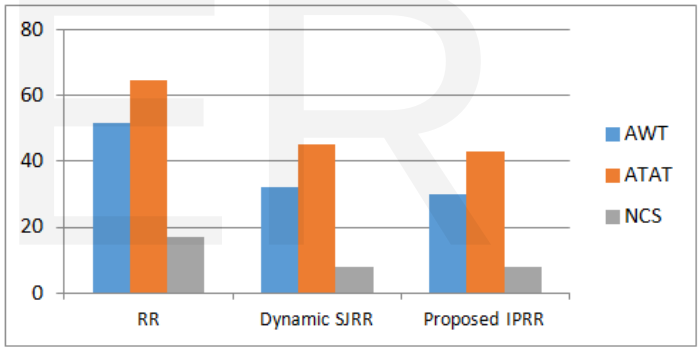

Fig. 11: Graphical representation of Table 4

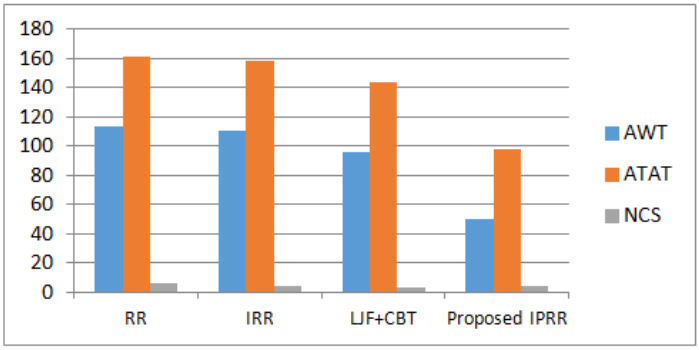

Fig. 12: Graphical representation of Table 6

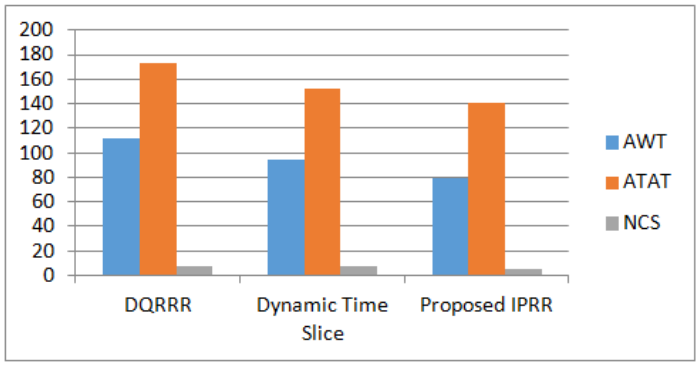

Fig. 13: Graphical representation of Table 8

From the result, it can be concluded that the proposed IPRR algorithm performs better by reducing AWT, ATAT and NCS.

7 Conclusion

The allocation of CPU to the processes is crutial in operating system. For this purpose, several CPU scheduling algorithms (FCFS, SJF, PS, RR etc.) already been introduced. But they have their own adavantages, workflow and performance issues (waiting time, number of context swiches etc). And the performance of CPU heavily depends on CPU scheduling algorithm. Several authors perform modification of existing algorithms which is superior in the context of performance. The proposed IPRR works as: firstly, it checks all the processes are in request queue or not. If no, then allocate CPU to the processes in RR fashion. Secondly, if all the processes are available in request queue then non-preemptive SJF is applied for the remaining processes including the remaining burst time of the running process. Hence, the performace of proposed IPRR algorithm performs better by reducing the AWT, ATAT and NCS.

In future, pipeline technology can be integrated with IPRR algorithm to improve the performance.

References

[1] R. K. Yadav, A. K. Mishra, N. Prakash, and H. Sharma, "An improved round robin scheduling algorithm for CPU scheduling," International Journal on Computer Science and Engineering, vol. 2, no. 04, pp. 1064–1066, 2010.

[2] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, Seventh Edition, 7th edition. Hoboken, NJ: John Wiley & Sons, 2004.

[3] A. K. Sain, "Dynamical Modified RR CPU Scheduling Algorithm," International Journal of Computer Trends and Technology (2231-2803), Volume, no. 4, pp. 90–93.

[4] 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, p. 1831, 2009.

[5] A. Noon, A. Kalakech, and S. Kadry, "A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average," arXiv preprint arXiv:1111.5348, 2011.

[6] A. P. M. Srivastav, S. Pandey, I. Gahoi, and N. K. Namdev, "Fair priority round robin with dynamic time quantum: FPRRDQ," International Journal of Modern Engineering Research (IJMER), vol. 2, pp. 876–881, 2012.

[7] M. K. Mishra, "An Improved Round Robin CPU Scheduling Algorithm," Journal of Global Research in Computer Science, vol. 3, no. 6, pp. 64–69, 2012.

[8] A. Abdulrahim, S. E Abdullahi, and J. B Sahalu, "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm," International Journal of Computer Applications, vol. 90, no. 4, pp. 27–33, 2014.

[9] M. Shoaib and M. Z. Farooqui, "A Comparative Review of CPU Scheduling Algorithms," in Proceedings of National Conference on Recent Trends in Parallel Computing (RTPC-2014).

[10] E. O. Oyetunji and A. E. Oluleye, "Performance assessment of some CPU scheduling algorithms," Research Journal of Information Technology, vol. 1, no. 1, pp. 22–26, 2009.

[11] M. A. H. Al-Hagery, "A selective quantum of time for round robin algorithm to increase CPU utilization," International Journal of Computer Information Systems, vol. 3, pp. 54–59, 2011.

[12] I. S. Rajput and D. Gupta, "A priority based round robin CPU scheduling algorithm for real time systems," International Journal of Innovations in Engineering and Technology, vol. 1, no. 3, pp. 1–11, 2012.

[13] H. S. Behera, R. Mohanty, and D. Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis," arXiv:1103.3831 [cs], Mar. 2011.

[14] L. Kishor, D. Goyal, R. Singh, and P. Sharma, "Optimized scheduling algorithm," in Proceedings of the IJCA International Conference on Computer Communication and Networks (CSICOMNET'11), pp. 130–134, 2011.

[15] S. Hiranwal and K. C. Roy, "Adaptive round robin scheduling using shortest burst approach based on smart time slice," International Journal of Computer Science and Communication, vol. 2, no. 2, pp. 319–323, 2011.

[16] I. Abdullahi and S. B. Junaidu, "Empirical Framework to Mitigate Problems in Longer Job First Scheduling Algorithm LJF+ CBT," International Journal of Computer Applications, vol. 75, no. 14, 2013.

[17] L. Datta, "Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice," IJ Education and Management Engineering, vol. 2, pp. 10–19, 2015.

About the Authors

Mehedi Raihan is currently working as a web and software developer at Kodesolution. Recently he has completed his B.Sc. in Computer Science and Engineering at North Western University, Khulna. Email: mehedismr@gmail.com.

Md. Zakaria Hossain is working as a web developer at Kodesolution. He has just completed his B.Sc. in Computer Science and Engineering at North Western University, Khulna. Email: jakahossain@gmail.com.

Md. Alam Hossain is currently serving as a Chairman, Department of Computer Science and Engineering (CSE), Jessore University of Science and Technology (JUST), Bangladesh, Email: alamcse_iu@yahoo.com.

Md. Rashid Al Asif is currently doing his M.Sc. Engg. in CSE at JUST, Email: rshid.al.asif@gmail.com. His is interested in the field of cloud computing, deep learning, algorithms, IoT and machine learning.

Md. Abdul Momin is an instructor in the field of CSE and interested in data structures, algorithms, image processing and bioinformatics. His E-mail: momin0625@gmail.com.