International Journal of Engineering & Technology, 7 (4.5) (2018) 238-241

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

An Improved Round Robin CPU Scheduling Algorithm based on Priority of Process

Govind Prasad Arya[1]*, Kumar Nilay[2], Devendra Prasad[3]

[1]Assistnat Professor , Sikkim Manipal University, Gangtok
[2]Kumar Nilay, Jawahar Navodaya Vidyalaya, Ernakulum, Kerla
[3]Assistant Professor, Poornima University, Jaipur
**Corresponding author E-mail:govind.arya10@gmail.com*

Abstract

The most important and integral part of a computer system is its operating system. Scheduling various resources is one of the most critical tasks an operating system needs to perform. Process scheduling being one of those tasks, involves various techniques that define how more than one processes can be executed simultaneously. The primary aim here is to the system more efficient and faster. The fundamental scheduling algorithms are: First Come First Serve (FCFS), Round Robin, Priority Based Scheduling, and Shortest Job First (SJF). This paper focuses on Round Robin Scheduling algorithm and various issues related to it. One major issue in RR scheduling is determining the length of Time Quantum. If the Time Quantum is too large RR scheduling behaves as FCFS. On the other hand, if it is too small it forces considerable increase in the number of context switches. Our main objective is to overcome this limitation of traditional RR scheduling algorithm and maximize CPU utilization, further, leading to more efficient and faster system. Here we propose an algorithm that categorizes available processes into High Priority processes and Low Priority process. The proposed algorithm reduces the average waiting time of High Priority processes in all cases and of Low Priority processes in not all but some cases. The overall waiting time changes on the basis of set of processes considered. The simulation results justify that the proposed schemes reduces the overall average waiting time when compared to the existing schemes.

Keywords: *CPU Scheduling, Round Robin Scheduling, Priority Scheduling, Waiting Time, Turnaround Time, Time Quantum*

1. Introduction

The technique through which a process currently having the control of CPU can be put on hold (in cases such as unavailability of resources like I/O) and allowing another process (in waiting state) to acquire the CPU control so as to make efficient CPU utilization is termed as CPU scheduling[11]. CPU scheduling is done with the primary objective of making the system more effective, efficient and justified and further maximizing CPU utilization.
Process scheduling in turn is the action performed by the process manager for detachment of the running process from the CPU and attachment of a waiting process to the CPU using certain detachment/attachment strategies. It is the art that resides at the core of Multiprogramming operating systems [12]. Loading of multiple processes into the executable memory is done by the operating system allowing them to share the CPU using time multiplexing.
CPU scheduling algorithms can be categorized into two categories namely, pre-emptive and non pre-emptive CPU scheduling. In pre-emptive CPU scheduling algorithms, a process currently acquiring the CPU can be forced to release the CPU control to any other process which is having higher priority and in doing so the current process goes into waiting state and the process with higher priority gains the control of CPU and starts executing [15]. More precisely the policy of permitting processes that are logically runnable to be temporarily suspended is called Preemptive Scheduling. However, if processes having higher priority constantly arrive in the ready queue then, the process having low priority suffers from starvation. The major limitation of pre-emptive scheduling algorithms is the overheads involved in scheduling the processes for pre-emption. In the non-pre-emptive category of scheduling, if a process has the CPU control then, by no means the CPU can be detached from that process until its execution is finished [16].
The most primitive CPU scheduling algorithms are; First in first out (FIFO), Shortest job first (SJF), Priority scheduling and Round robin CPU scheduling algorithm.

2. Literature Survey

The most preliminary CPU scheduling is FCFS in which jobs/processes are allocated the CPU on First Come First Serve basis [1]. It is easy to understand and can be implemented using FIFO queue in non pre-emptive as well as pre-emptive mode depending upon the requirements. However, it suffers from the major limitation of presenting higher Average Waiting Time of processes.
The second CPU scheduling algorithm is SJF with the principle of executing the shortest job first [3]. SJF also can be implemented in non pre-emptive as well as pre-emptive mode. It is considered as the best CPU scheduling technique for reducing the waiting time. Its implementation is best suited to batch systems where the required CPU time is known prior to execution. However, it doesn't work very well with interactive systems where CPU time is not known in advance. In other word, the primary requirement for SJF scheduling is the prior knowledge of execution time of processes.
Priority Scheduling is essentially a pre-emptive scheduling technique which is most prevalent in batch systems [5]. Here, each

process is allotted a priority and the process with highest priority is executed firstly, then the process having second highest priority, henceforth and so on. In case the processes having equal priority arrive, then FCFS is used to resolve the conflict. Priority is assigned to the processes basis the memory requirements, time requirements or any other resource requirements.
The other CPU scheduling technique is Round Robin which also is essentially pre-emptive. Here, CPU is allocated to the processes in a circular fashion for a fixed time period called Time Quantum [8].
When this Time Quantum is reduced to zero, it is preempted and other process start it's execution for a given time period. The status of pre-empted processes is saved by context switching method.
One other variant of CPU scheduling is the Multiple-level queues which a manual scheduling technique [15]. It groups the processes having mutually similar characteristics together using the other existing algorithms. A number of queues are retained for processes with mutual characteristics. Each queue can have its specific scheduling algorithms [8]. Each queue is also assigned a priority for execution. For instance, OS-bound jobs can be arranged in one queue and all I/O-bound jobs in another queue. The Process Scheduler then in turn selects jobs from each queue and allots them to the CPU based on the algorithm allotted to the queue. Multi-level queue scheduling was generated for circumstances in which processes are certainly categorized into different groups.

3. Shortcomings of Existing Algorithm

We considered traditional round robin CPU scheduling algorithm as existing algorithm. Although round robin is an efficient CPU scheduling algorithm because it treat all the processes equally and provide equal chance to execute. As per the research, it is found that there are some critical processes in the system having high priority. Therefore traditional round robin algorithm does not meet the expectations at this condition.
Consider the following set of processes with time quantum 4 –

Table 1 Processes in Ready Queue for Existing Methodology

Process Name	Priority	Burst Time
P0	0	5
P1	1	3
P2	1	12
P3	0	9
P4	0	8

We know that round robin scheduling provides equal opportunity to execute all the processes in the process set. Hence, the Gantt chart and waiting time for the above set of processes are shown in figure 1
.

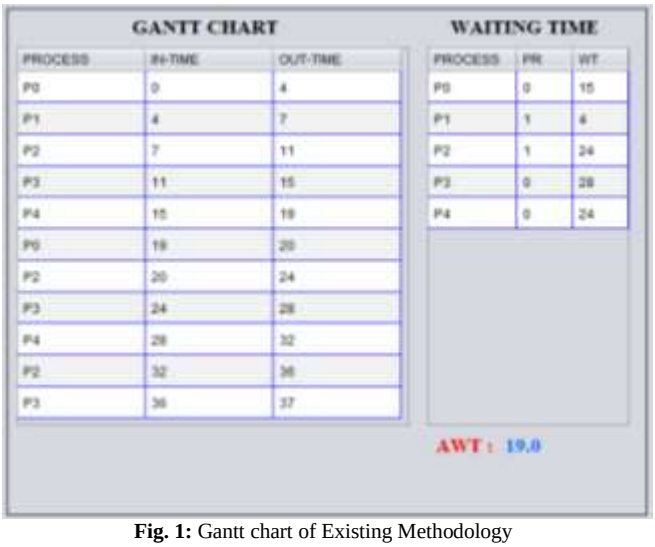

Fig. 1: Gantt chart of Existing Methodology

The average waiting time (AWT) of low and high priority processes is shown below in figure 2.

AWT	Existing
AWT of LPQ	22.333334
AWT of HPQ	14.0
Overall AWT	19.0

Fig. 2: Waiting Time Analysis of Existing Methodology

4. Proposed Method

The Round Robin (RR) CPU scheduling algorithm considers all the jobs, which are present in ready queue as equal priority jobs. It offers identical chance to all the jobs to execute in CPU for duration called Time Quantum (TQ). So, a process can be executed by the CPU until its time quantum (TQ) terminates or the process terminates by its own after conclusion of its CPU worst time.
It may be probable that processes present in ready queue are of diverse priority i.e. high or low. Some processes may need Central Processing Unit on urgent basis (i.e. program to shut down computer because of temperature exceeded, alert on unauthorized access, etc.) are called high priority jobs. On the other hand, another type of processes is with normal priority.

5. Proposed Algorithm

Our proposed algorithm is given below-

Step 1: Enter process name, priority and burst time.
Step 2: Store the above details in a queue called READYQ
Step 3: Create two separate queues, first HIGHPQ for high priority processes and second LOWPQ for normal priority processes.
Step 4: Do step 5 to step 11 until remaining CPU burst time of processes of both the queues (HIGHPQ and LOWPQ) become zero.
Step 5: Select next process from either HIGHPQ or LOPQ on alternate basis. First, a process form HIGHPQ must be selected as it should get priority over normal priority processes.
Step 6: If the remaining CPU burst time of selected process is greater than or equal to time quantum then do step 7, otherwise do step 8.
Step 7: Execute that process for a duration of time quantum.
Step 8: Execute the selected process until its remaining burst time become zero.
Step 9: Updated the remaining CPU burst time of the respective process in respective queue.
Step 10: Store the IN-TIME and OUT-TIME of the process into a table GANTTCHART.
Step 11: If above process has selected from HIGHPQ then swap the next turn to LOWPQ and vice versa.
The high priority processes should get priority to execute. In this research, I have proposed a methodology, which provide alternate chance to high and low priority processes. A process from high priority queue is selected first then next process is selected from low priority queue. The steps for the methodology are given below-

HIGHPQ: This queue contains the processes of high priority.

Process Name	Priority	Burst Time
P1	1	3
P2	1	12

LOWQPQ: This queue contains the processes of low priority.

Process Name	Priority	Burst Time
P0	0	5
P3	0	9
P4	0	8

The Gantt chart and waiting time for the processes of table 1 with time quantum 4 is shown below in figure 3.

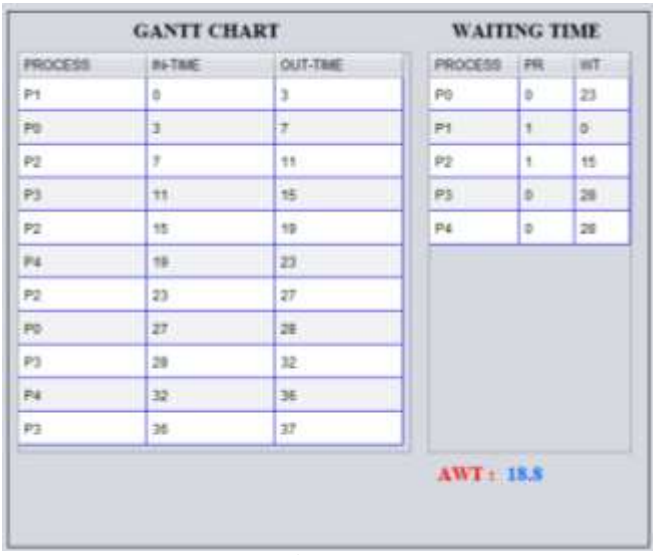

Fig.3: Working of Proposed Methodology

6. Result and Analysis

As we can see in the below shown figure 4, the average waiting time of high priority process is 7.5 using proposed algorithm which is approximately half of average waiting time (14.0) using existing algorithm. The overall waiting time also better using proposed algorithm.

AWT	Existing	Proposed
AWT of LPQ	22.333334	26.333334
AWT of HPQ	14.0	7.5
Overall AWT	19.0	18.8

Fig. 4: Result Analysis of Existing Vs Proposed Methodology

The same result can be analyzed using bar chart shown in figure 5.

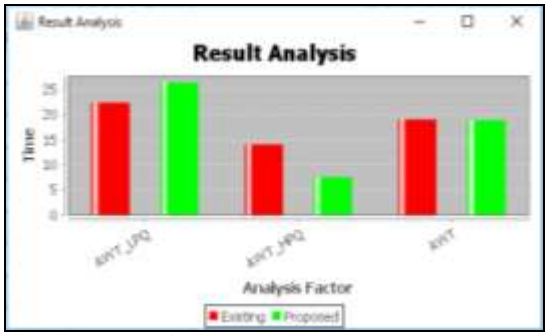

Fig. 5: Result Analysis of Existing Vs Proposed Methodology Using Bar chart

7. Conclusion

In this research, I have preserved the motivation of traditional round robin that all process should get chance to execute after a time quantum. The only improvement is that if high priority process are stored at rear side in the ready queue, then those processes will not bounded to execute too late due to late arrival. The proposed methodology will definitely reduce average waiting time of high priority processes however; it may increase the average waiting time of normal priority processes. The overall average waiting time of all the processes stored in ready queue may or may not improve depending on set of processes. Although the proposed algorithm shows better result for high priority processes, still there is always a need and motivation for better results. In future, the result can be improved using variable time quantum. The execution of algorithm can also be improved by using efficient data structures.

References

[1] Sanjay Kumar Panda and Saurav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure", International Journal on Computer Science and Engineering, 4(1), pp. 45-53, January 2012.

[2] Pallab Banerjee, Probal Banerjee, Shweta Sonali Dhal, "Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm usingStatic Time Quanmtum", International Journal of Innovative Technology and Exploring Engineering, 1(3), pp. 56-62,August 2012.

[3] P.Surendra Varma, "A Finest Time Quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) Algorithm", Journal of Global Research in Computer Science, 4 (3), pp. 10-15, March 2013.

[4] Raman, Dr.Pradeep Kumar Mittal, "An Efficient Dynamic Round Robin CPU Scheduling Algorithm (EDRR)", International Journal of Advanced Research in Computer Science and Software Engineering, 4(5), pp. 907-910, May 2014.

[5] Silberschatz, A., P.B. Galvin and G. Gagne, Operating Systems Concepts. 7th Edn., John Wiley and Sons, USA., ISBN:13: 978-0471694663, pp. 944.

[6] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", Proc. of International Symposium on Computer Engineering & Technology 2010, Vol 17, pp. 126-137, 2010 .

[7] R. J. Matarneh, "Seif-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Proceses", American Journal of Applied Sciences, 6(10), pp. 1831-1837, 2009.

[8] 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, 5(5), pp. 10-15, August 2010.

[9] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", Sixth Edition.

[10] E.O. Oyetunji, A. E. Oluleye," Performance Assessment of Some CPU Scheduling Algorithms", Research Journal of Information Technology,1(1): pp 22-26, 2009

[11] Ajit Singh, Priyanka Goyal, Sahil Batra," An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 07, 2383-2385, 2010.

[12] Rami J. Matarneh."Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences 6(10):1831-1837, 2009.

[13] Sourav Kumar Bhoi, Sanjaya Kumar Panda, Debashee Tarai, "Enhancing cpu performance using subcontrary mean dynamic round robin (smdrr) scheduling algorithm" ,JGRCS, Volume 2, No. 12, December 2011, pp.17-21

[14] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", International Symposium on Computer Engineering & Technology (ISCET), Vol 17, 2010

[15] P.Surendra Varma , "A FINEST TIME QUANTUM FOR IMPROVING SHORTEST REMAINING BURST ROUND ROBIN (SRBRR) ALGORITHM" Journal of Global Research in Computer Science, 4 (3), March 2013, 10-15
[16] Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash, Himanshu Sharma," An Improved Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 1064-1066, 2010
[17] Ishwari Singh Rajput," A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems", (IJIET)International Journal of Innovations in Engineering and Technology Vol. 1 Issue 3 Oct 2012
[18] Manish Kumar Mishra & Abdul Kadir Khan, (2012) "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, Vol. 3, No. 6, pp 64-69.
[19] Abdulrazak Abdulrahim, Salisu Aliyu, Ahmad M Mustapha & Saleh E. Abdullahi, (2014) "An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 2, pp 601-610.
[20] Abdulrazak Abdulrahim, Saleh E. Abdullahi & Junaidu B. Sahalu, (2014) "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm", International Journal of Computer Applications, Vol. 90, No. 4, pp 27-33.
[21] An Effective Round Robin Algorithm using Min-Max Dispersion Measure. Panda, Sanjaya Kumar; Bhoi, Sourav Kumar // International Journal on Computer Science & Engineering;Jan2012, Vol. 4 Issue 1, p45
[22] Designing Various CPU Scheduling Techniques using SCILAB. Saini, Mona // International Journal of Computer Science & Information Technolo;2014, Vol. 5 Issue 3, p2918
[23] Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes. Matarneh, Rami J. // American Journal of Applied Sciences;2009, Vol. 6 Issue 10, p1831
[24] Two Queue based Round Robin Scheduling Algorithm for CPU Scheduling. Jindal, Srishty; Grover, Priyanka // International Journal of Computer Applications;Nov2014, Vol. 105 Issue 1-18, p21
[25] A 2LFQ Scheduling with Dynamic Time Quantum using Mean Average. Lenka, Rakesh K.; Ranjan, Prabhat // International Journal of Computer Applications;6/1/2012, Vol. 47, p15
[26] Improvised Round Robin (CPU) Scheduling Algorithm. Sirohi, Abhishek; Pratap, Aseem; Aggarwal, Mayank // International Journal of Computer Applications;Aug2014, Vol. 99, p40
[27] A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average. Noon, Abbas; Kalakech, Ali; Kadry, Seifedine // International Journal of Computer Science Issues (IJCSI);May2011, Vol. 8 Issue 3, p224