RR Scheduling Algorithm Prioritizing Shortest Time with Incrementing Time Quantum on Burst Time

Austria, Kathy Dumacil, Jamellah Malicdem, Tricia Mae
Villagracia, Jolie Janelle Baquirin, Rey Benjamin

Abstract:

Process management is one of the important tasks performed by the operating system. The performance of the system depends on the CPU scheduling algorithms. The main aim of the CPU scheduling algorithms is to minimize waiting time, turnaround time, response time and context switching and maximizing CPU utilization. First-Come-First-Served (FCFS) Round Robin (RR), Shortest Job First (SJF) and, Priority Scheduling are some popular CPU scheduling algorithms. In time shared systems, Round Robin CPU scheduling is the preferred choice. In Round Robin CPU scheduling, performance of the system depends on the choice of the optimal time quantum. This paper presents an improved Round Robin CPU scheduling algorithm coined enhancing CPU performance using the features of Shortest Job First and Round Robin scheduling with incrementing time quantum. The proposed algorithm is experimentally proven better than conventional RR. The simulation results show that the waiting time and turnaround time have been reduced in the proposed algorithm compared to traditional RR.

I. INTRODUCTION

In computer science, scheduling is the procedure by which processes are given access to system resources (e.g. Processor cycles, communications bandwidth). The need for a scheduling algorithm arises from the requirement of fast computer systems to perform multitasking, which is to execute more than one process at a time, and multiplexing, or the act of transmitting multiple flows simultaneously.

Scheduling is a fundamental operating system function that determines which process to perform, when there are multiple runnable processes. CPU scheduling is important because it greatly influences resource utilization and other performance parameters. There exists a number of CPU scheduling algorithms such as: First Come First Serve, Shortest Job First Scheduling, Round Robin scheduling, Priority Scheduling etc, but due to a number of disadvantages, these are rarely used in real time operating systems except for the previously stated: Round Robin scheduling.

According to Noon, A., Kalakech, A., Kadry, S. (2011), Round Robin is considered as the most widely adopted CPU scheduling algorithm, and is one of the oldest, simplest, fairest and most widely used scheduling algorithms. It is designed particularly for time-sharing systems and undergoes severe problems directly related to quantum size.

In RR a small unit of time is used which is referred to as Time Quantum" or Time slice. In RR, If the time quantum chosen is too large, the response time of the processes is considered too high. On the other hand, if the time quantum is too small, it increases the overhead of the CPU.

The CPU scheduler goes around the Ready Queue allocating the CPU to each process for a time interval up to 1 time quantum. If a process's CPU burst exceeds 1 time quantum, that process is preempted and is put back in the ready queue. If a new process arrives then it is added to the tail of the circular queue. Out of the previously mentioned algorithms, RR provides better performance as compared to the others in case of a time sharing operating system. The performance of a scheduling algorithm depends upon the scheduling criteria viz. Turnaround time, Waiting time, Response time, CPU utilization, and Throughput.

In this paper, the researchers aim to propose an algorithm that involves the RR (Round Robin) Algorithm prioritizing shortest time with incrementing time quantum on burst time. This algorithm will execute the shortest job first and give an increasing time quantum accordingly to the following tasks in queue.

II. REVIEW OF RELATED LITERATURE

In every CPU scheduling algorithm consists of different properties, in our proposed algorithm, to be able to conclude the computed time quantum on burst times. Many criteria have been suggested for comparing CPU scheduling algorithm. The following criteria include:

CPU UTILIZATION: the percentage of time CPU remains busy.

TURNAROUND TIME: the submission of processing time to its completion time. It refers to the total time spent waiting in the Ready Queue to be executed.

THROUGHPUT: the number of processes completed per unit time.

WAITING TIME: the amount or sum of processing time spent waiting in the Ready Queue.

CONTEXT SWITCH: the storing previous processes, so that the stored processes can be restored and execution is/are resumed later. It is an essential feature of a multitasking operating system.

RESPONSE TIME: the time from the submission of a process until the first response.

Users must consider the characteristics of these algorithms to ensure better performance metrics. In recent times, different approaches are used to increase the performance of CPU scheduling algorithm. Rakesh Kumar Yadav et al. [4] utilised the concept of SJF in RR algorithm. Ajit Singh et al. [5] combined the concept of SJF in RR algorithm. After each cycle they double the time quantum. Manish Kumar Mishra et al. [6] also merged the concept of Shortest Job First (SJF) with Round Robin (RR) to minimize the waiting time & turnaround time. After each complete cycle they chose the burst time of shortest process as the new time quantum. Rishi Verma [7] calculated the time quantum after every cycle by

subtracting the minimum burst time from maximum burst time. Neetu Goel et al. [8] took two dynamic numbers K (as its time quantum) and F. During the execution, it was checked if the remaining burst time of process in execution was less than time-quantum/F, then the process would continue its execution otherwise the process would stop its execution and would go to the end of the ready queue. M. Ramakrishna et al. [9] added the concept of priority scheduling in RR scheduling to optimize the Round Robin scheduling.

Rami J. Matarneh [9] proposed an algorithm SARR to improve the performance of Round Robin. In SARR for each cycle the median of burst time of the processes is calculated and used as time quantum. H.S.Behera et al. [11] also used a similar type of algorithm. But they again rearranged
the processes during their execution. Their algorithm selects the process with the lowest burst time, then proceeds to implement the process with the highest burst time, then continues with the process that has the second lowest burst time, and so on.

III. METHODOLOGY

In this study, the researchers are aiming to enhance and improve the operating system/s' scheduling algorithm, specifically the combination of the Shortest Job First(SJF) and Round Robin(RR). Through past research and analysis, Round Robin, being a well-recognized scheduling algorithm, falls under its lack to assign what to prioritize first and takes time for processing with such limited time quantum. That is why a proposal of modifying the Round Robin is made, prioritizing shortest time for faster and more efficient run on an operating system, with the addition of incrementation to its time quantum in order to aid and lessen context switching. The algorithm proposed consists of 10 to 11 steps in total.

PSEUDOCODE:

1. Start the process
2. Declare the number of process where n= number of process
3. Take number of elements to be inserted
4. All the processes present (on BT) are stored on the remaining time (rem)
5. Look for the maxBT and the minBT from inputted Burst Times
   temp=(MAXBT + MINBT)/n
   TQ = temp + [(temp/2)*i)
   /*TQ= Time Quantum; maxBT= maximum Burst Time; minBT= minimum Burst Time; temp= Average of Minimum and Maximum Burst Times
   If one process is there the TQ == BT of itself */
6. Process are sorted according to BT's in ascending order where shortest time will be prioritized first
7. //assign TQ to (1 to n) process
   For i = 1 to n
   $\{P_1 \rightarrow \text{TQnew}\}$
   End for

//assign TQ_{new} to the available processes as the loop continues, thus adding an incrementation every repetition

8. Calculate the remaining burst time of the processes
9. if (new process arrived and BT !=0)
   Go to step 4
   else if (new process did not arrive and BT!= 0)
   Go to step 5
   else if (new process arrived and BT ==0)
   Go to step 4
   else
   Go to step 10
   end if
   end while
10. Calculate ATT, AWT and CS
    /* where ATT = Average Turnaround Time, AWT = Average Waiting Time,
    CS = number of Context Switches
11. End

IV. ILLUSTRATION

For a more specific illustration of how the RRSJF or Round Robin prioritising shortest job first scheduling algorithm works, the researchers enter 5 as the number of processes needed, p1, p2, p3, p4, and p5. Table 4 shows that there is no need for burst time to arrive or entered in order for once all of the processes are completed, a sorting will happen automatically, from ascending order. Once sorted, maximum and minimum burst time will be identified or searched and will be computed according to the formula derived from the given pseudocode, which is to get the quantum time using these maximum and minimum burst time, along with the number of processes. In the given table, 6 is the minimum burst time and 10 is the maximum burst time, arriving at 4 as the quantum time once step 5 of the pseudocode is executed. The only thing that the code accepts are integers. Once done, gantt chart is displayed as a basis for the total amount of context switches for it counts how long did it take for every process to be computed. In the table, p1 is assigned to 10 as its burst time, but later on changed into p5 for it was rearranged when sorted, same goes for the other process where p2 was once 8, and was changed to p3 and so on. When waiting time and average time were computed, they were executed according to priority, where shortest time comes first before the other. Once all are executed and no more remaining burst times, average waiting time and average turnaround time are displayed. As a result, there is a slight difference that changed in terms of the average waiting time and average turnaround time. Compared to the round robin, average waiting time is 26 and the average turnaround time is 34. But when sorted and shortest time was prioritized, average waiting time decreased by 20, making it just 6 and average turnaround time decrease by 20 also, making it just 14.

V. EXPERIMENTAL ANALYSIS

Assumptions

For performance evaluation, it has been assumed that all the processes are having equal priority in a single processor environment. The number of processes and their burst time are known before submitting the processes for the execution. The context switching overhead incurred in switching from one process to another has been considered zero. The overhead of arranging the ready queue processes in ascending order has also been considered zero. All processes are CPU bound. No processes are I/O bound. The time quantum is taken in milliseconds (ms).

Test Cases

Test case scenarios will be performed to compare the different algorithms based on the following priorities:

1. Turn around time
2. Context switch
3. Waiting time

Table 1: A test case showing the results processed with the use of the round robin scheduling algorithm where the first inputted burst time are prioritized first.

Process	Burst	Waiting	Turn Around
p1	**10**	**30**	**40**
p2	**8**	**27**	**35**
p3	**7**	**27**	**34**
p4	**6**	**21**	**27**
p5	**9**	**27**	**36**

Quantum time: 3
Gantt chart: p1-p2-p3-p4-p5-p1-p2-p3-p4-p5-p1-p2-p3-p5-p1
Ave. Waiting Time: 26
Ave. Turn Around Time: 34
CS: 14

Table 2: A test case showing the results processed with the use of the shortest job first scheduling algorithm where the priority is the shortest or smallest burst time, but needs an input for arrival time.

Process	Arrival	Burst	Waiting	Turn Around
p1	**0**	**10**	**30**	**40**
p2	**0**	**8**	**13**	**21**
p3	**0**	**7**	**6**	**13**
p4	**0**	**6**	**0**	**6**
p5	**0**	**9**	**21**	**30**

Quantum time: 3
Gantt chart: n/a
Ave. Waiting Time: 14
Ave. Turn Around Time: 22

Table 3: A test case showing the combination of Round Robin and Shortest Job first.

Process	Burst	Priority	Waiting	Turn Around
p1	**10**	**5**	**24**	**34**
p2	**8**	**3**	**16**	**24**
p3	**7**	**2**	**16**	**23**
p4	**6**	**1**	**16**	**22**
p5	**9**	**4**	**24**	**33**

Quantum time: 4
Ascending order:

p1: 6	**p2: 7**	**p3:8**	**p4:9**	**p5:10**

Gantt chart: p1-p2-p3-p4-p5-p1-p2-p3-p4-p5-p4-p5
Ave. Waiting Time: 19
Ave. Turn Around Time: 27
CS: 11

Table 3: A test case showing the results of table 3. In addition, priority for executing process is by shortest time first, finding time quantum is changed using the given pseudocode.

Process	Burst	Priority	Waiting	Turn Around
p1	10	5	16	26
p2	8	3	0	8
p3	7	2	0	7
p4	6	1	0	6
p5	9	4	16	25

Quantum time: 8
Ascending order:

p1: 6	p2: 7	p3:8	p4:9	p5:10

Gantt chart: p1-p2-p3-p4-p5-p4-p5
Ave. Waiting Time: 6
Ave. Turn Around Time: 14
CS: 6

Results

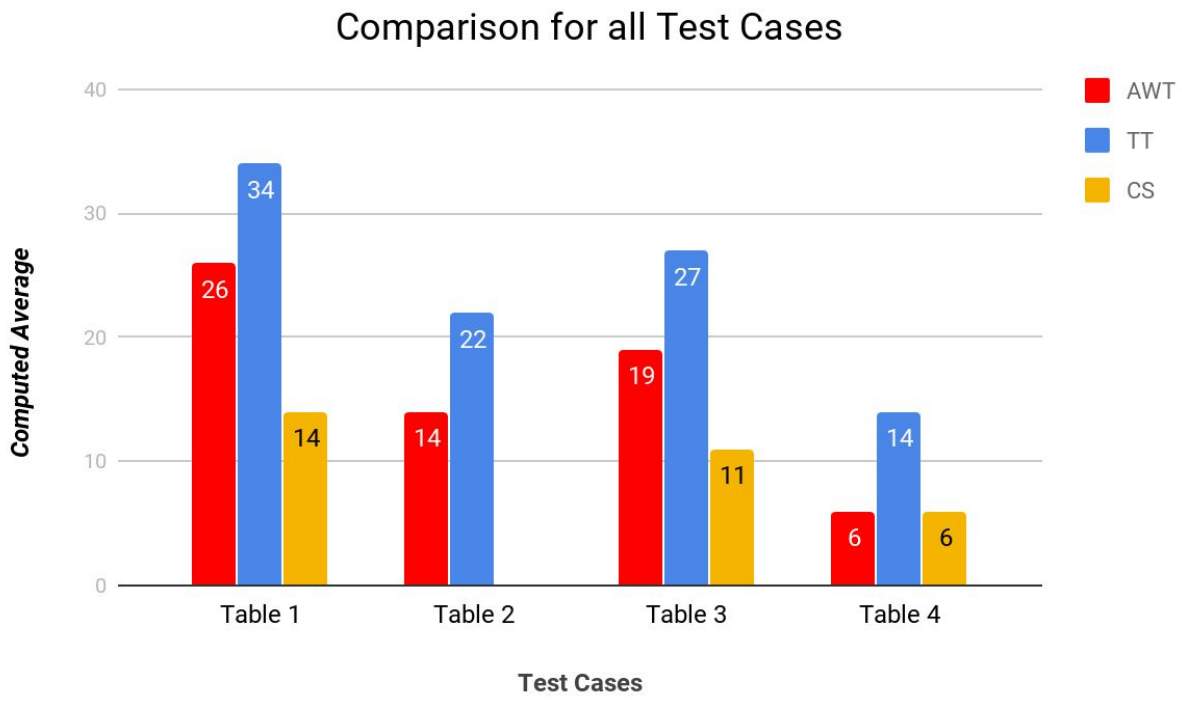

Chart 1: Comparison for all Test cases. In terms of Average waiting time, Table 1 (Round Robin) has the highest computed average while Table 4 (our modified Round RObin prioritizing shortest job first) has the lowest. While in Average Turnaround time, Table 1 (Round Robin) has the highest computed average, while Table 4 (Shortest Job First) is the lowest. But in terms of the computed context switches, there is no recorded context switch for Table 2 (Shortest Job First) and Table 4 (our modified Round Robin prioritizing shortest job first) has the lowest context switch excluding Table 2.

VI. CONCLUSIONS

One of the important tasks of the operating system is the allocation of CPU to the processes waiting for execution. Many CPU scheduling algorithms have been presented with some advantages and disadvantages. An improved round robin CPU scheduling algorithm with varying time quantum with the use of incrementation proposed in this paper giving better performance than conventional RR algorithm. The waiting time and turnaround time have been reduced in the proposed RRSJF scheduling algorithm and hence the system performance has been improved. Simulation results also prove the correctness of the theoretical results. The proposed algorithm can be integrated to improve the performance of the systems.

VII. RECOMMENDATIONS

Based on the results of the study, the following are therefore recommended:

1. To improve the accuracy of the algorithm, it is necessary to conduct a more thorough exploration
2. It is recommended that the test case strategies employed could be further expounded and be altered to test the validity of the outputs.
3. Research related to scheduling algorithms that provide a means of defining their succession and enhancement would be of value to the field of Computer Science in the society.
4. Furthermore, the researchers recommend further research and more diverse approaches for the future studies to be conducted.

References:

[1]Baquirin, R.B., Guevara, K.S., Manuel, J.I. and Tandingan, D. (2019). Fittest Job First Dynamic Round Robin (FJFDRR) Scheduling Algorithm Using Dual Queue and Arrival Time Factor: A Comparison. IOP Conference Series: Materials Science and Engineering. 10.1088/1757-899X/482/1/012046. Vol. 482

[2]Singh, A., Goyal, P., Batra, S. (2010). An Optimized Round Robin Scheduling Algorithm for CPU Scheduling. (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 07, 2010, 2383-2385. Retrieved from https://pdfs.semanticscholar.org/780d/07fbf09484a2490868d45

255a23762a7e21d.pdf) on September, 2019

[3]Noon, A., Kalakech, A., Kadry, S. (2011). A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average. IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, 2011, 224-229. Retrieved from https://arxiv.org/abs/1111.5348 on September, 2019

[4]Nayak, D., Malla, S.K., Debadarshini, D.(2012). Improved Round Robin Scheduling using Dynamic Time Quantum. International Journal of Computer Applications (0975 – 8887) Volume 38– No.5. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.4815&rep=rep1&type=pdf on September, 2019

[5]Nosrati, M., Karimi R., Harriri, M. (2012). Task Scheduling Algorithms Introduction. World Applied Programming, Vol (2), Issue (6), 394-398. ISSN: 2222-2510. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.683.7296&rep=rep1&type=pdf on September, 2019

[6]Hiranwal, S., & Roy, K.C. (2011). Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice. International Journal of Computer Science and Communication, 2(2), 319-323.

[7]Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash, Himanshu Sharma (2010) "An Improved Round Robin Scheduling Algorithm for CPU Scheduling", International Journal on Computer Science and Engineering, pp 1064-1066.

[8]M. Ramakrishna, G. Pattabhi Rama Rao (2013) "Efficient Round Robin CPU Scheduling Algorithm for Operating Systems", International Journal of Innovative Technology and Research, pp 103-109.

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

[10]H.S.Behera, R. Mohanty, Debashree Nayak, (2010) "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance", International Journal of Computer Applications, pp 10-15.