Modifying Round Robin Algorithm for Process Scheduling using Dynamic Quantum Precision

Mohd Abdul Ahad
Department of Computer Science, FMIT, Jamia Hamdard, New Delhi, India

ABSTRACT

Any processor is said to be optimally performing if it has a higher throughput, low waiting time, low turn around time, and less number of context switches for the processes coming for execution. The scheduler is the component which is responsible for taking the decision on how and when to schedule the processes waiting in the ready queue for their chance to get the CPU. In this paper we have given a proposal for modifying the classical round robin scheduling algorithm. The proposal works by finding the left out burst time of the processes in the last but one turns of the quantum cycle. The left out burst time values in last but one turn of each process are processed to get an optimal threshold value. Based on this, we have divided the processes waiting in the ready queue in two categories. The processes in the first category are the one for which we will modify the time quantum and the processes in the second category will be processed as per the classical round robin algorithm. We have also compared our proposal with the classical round robin algorithm and the results are depicted in tabular form.

Keywords: Scheduling, Turn around Time, Context Switches, waiting time.

1. INTRODUCTION

In today's era, most processors are capable of handling multiple processes at the same time by switching the allotment of the CPU among the processes. This switching is performed by the scheduler of the operating system. The competence of the processor depends upon the type of scheduling algorithm followed by the processor [9], [14]. There exists a variety of process scheduling algorithms and most of the operating systems are deploying more than one process scheduling algorithm for optimizing their processing needs[12], [13]. The three most common scheduling algorithms are as follows [10], [11]:

A. First Come First Serve (FCFS):

Here the CPU is assigned to the job or process that arrives first at the ready queue. The CPU is taken away from the process only when, it has totally finished its execution or is performing any I/O operation(s) [10], [11].

B. Shortest Job First (SJF):

In this the process that has the minimum burst time, gets the CPU first. In this algorithm also, the CPU is taken away only when the process has entirely completed its execution [10], [11], [12].

C. Round Robin (RR):

This algorithm is an enhanced version of the FCFS algorithm. It introduces the concept of time quantum (a fixed amount of time period). If the process entirely finishes its execution within this time, it is removed from the ready queue otherwise it goes back to the ready queue and waits for its next chance in FCFS order only [10], [11], [12].

Out of the above stated algorithms, the round robin algorithm is the most interesting because it involves the concept of quantum slices of time and context switching and this is the reason why we have chosen this algorithm for modification.

2. RELATED WORKS

There have been many researches going in the field of modifying the scheduling algorithm of the operating system so as to make them faster and efficient. The authors of [1] proposed a new method using integer programming for finding the value of time quantum [1]. The authors of in [2] proposed an adaptive version of the round robin algorithm wherein the concept of smart time slices is used by rearranging the processes in increasing order of the burst times. The novelist of [3] highlights the problem in the conventional Linux SCHED_RR, wherein the users who runs more processes will get more share of CPU than the users who runs less number of processes of the same priority. They proposed a solution to this by assigning equal share of CPU to users instead of the processes [3]. The researchers in [4] proposed an approach in which the process are arranged in ascending order of the burst time and then an optimal time quantum is calculated using the median concept, which means that if the number of processes in the ready queue is odd, the burst time of the middle process will become the time quantum, otherwise the average of the two middle processes will become the time quantum. The authors in [5] highlight an improved version of the algorithm with the complexity of O (n log n). They used the concept of bursts arrivals and batch departures [5]. The researchers in [6] propose the concept of self adjustment time quantum, in which the time quantum is continuously modified according to the burst time of the currently running processes [6]. The authors in [7] proposed to increase the time quantum of the process which does not completes its execution within the allotted time [7]. The authors in [8] talks about increasing the time quantum two folds after completing one cycle of CPU allocation to the processes. Then finding the process with the minimum execution time and assigning the CPU to it and continuing in the same fashion. The researchers in [15] calculates the time quantum using max-min approach wherein the time quantum is the difference between the maximum and minimum burst time value of the processes.

3. IMPROVED ROUND ROBIN APPROACH

Our approach is not to change the round robin philosophy but to make it one step advanced. In round robin approach the time quantum is absolutely fixed. While it is observed that in many cases jobs are preempted even if a negligible amount of execution time is left for a job. This particular job now has to wait for its next turn. This situation caters to waiting time of the process sometimes unnecessarily.

A. Terminologies Used in the proposed algorithm

- P: Process
- TQ: Time Quantum

- Old_TQ: A quantity storing the original initial value of the Time Quantum.
- BT: An N dimensional array storing the Burst Time of the processes
- AT: An N dimensional array storing the Arrival Time of the processes
- N: Total number of processes in the ready queue
- Count: An N dimensional array storing the value of the current quantum cycle of the processes.
- B: An N dimensional Array Storing the integer part of the quantity:

  B [P_i] = int (BT [Pi] / TQ), where i= 1,2, 3....N.
- Left out Time (LoT) : It is also an 'N' dimensional array that can be calculated by the equation:
- LoT [Pi] = BT [Pi] % TQ, where i = 1, 2, 3 ...N
- K : ceil (Average of (LoT[Pi]))

B. Proposed Modification of the Round Robin Algorithm

1. **Input:** Ready queue consisting of the various processes waiting to get the CPU
2. **For** each process in the ready queue do
3. Count [Pi] =1;
4. **If** (BT [Pi] < TQ)
5. {
6. Apply classical round robin policy
7. }
8. **If** (BT [Pi] / TQ) = = I), *where I is an Integer value*
9. {
10. Apply classical round robin policy
11. BT [Pi] = BT [Pi] -TQ;
12. }
13. **Else if** (BT [Pi]/ TQ) != I) *// Non integer value*
14. {
15. **If** (B [P_i] = = Count [P_i]) *// checking the last but one turn*
16. {
17. **if** (LoT[i] <=K)
18. {
19. TQ=TQ + K
20. Assign CPU to the process as per New TQ
21. BT [Pi] = BT [Pi] - TQ
22. TQ=Old_TQ;
23. }
24. **Else**
25. {
26. Apply classical round robin policy
27. BT [Pi] = BT [Pi] – TQ
28. }
29. }
30. **Else** (Count [P_i]! =B [P])
31. {
32. BT [Pi] = BT [Pi] –TQ;
33. Count [Pi] ++
34. }
35. }

C. Working of the Proposed Algorithm using hypothetical Examples

Method for calculating the threshold value (K)

1. Let BT[i] represents the Burst Time of the i^{th} process.
2. Let TQ represents the Time Quantum
3. Let 'n' is the number of processes in the ready queue.
4. Therefore the Left out Time (LoT) for a process is calculated as follows
   a. LoT[i] = BT[i] % TQ;

And the threshold value K is calculated as follows

K* = ceil (LoT [i] / n);

Where ceil (p) is the smallest integer greater than or equal to p.

* We will not include those processes in the calculation of 'K' for which LoT[i] < = 0;

Example 1:

TABLE 1: (Example 1 Using Conventional Round Robin Approach)

TIME QUANTUM (TQ) = 20		
Process Name	**Arrival Time**	**Burst Time**
P1	0	24
P2	0	29
P3	0	35
P4	0	40
P5	0	27

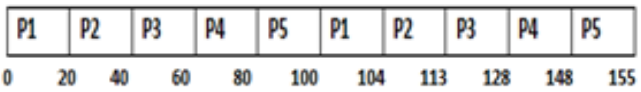

Fig 1: Gantt chart according to Conventional Round Robin Approach

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for Conventional Round Robin Approach

Average Turn around Time (Avg. TAT) is given by Avg. TAT = (104 + 113 + 128 + 148 + 155) / 5

= 129.6

Average Waiting Time (Avg. WT) is given by

Avg. WT = (80 + 84 + 93 + 108 + 128) /5

= 98.6

Number of Context Switches = 09

TABLE 2: Example 1(Using Proposed Approach)

TIME QUANTUM (TQ) = 20				
Process Name	**Arrival Time**	**Burst Time**	**B = INT(BT / TQ)**	**C = BT % TQ**
P1	0	24	1	4
P2	0	29	1	9
P3	0	35	2	15
P4	0	40	2	0
P5	0	27	1	7

Calculating the value of K as follows

K = ceil (Average of (C[i]))

K= ceil ((4 +9+15+7) / 4)

K= ceil (35/4) = ceil (8.75)

Therefore K = 9

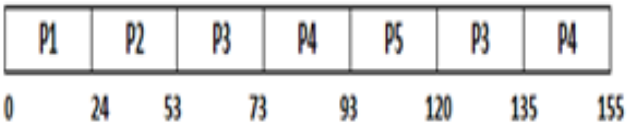

Fig 2: Gantt chart according to Proposed Approach

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for proposed Approach

Average Turn around Time (Avg.TAT) is given by

Avg. TAT = (24 + 53 + 135 + 155 + 120) / 5

= 97.4

Average Waiting Time (Avg. WT) is given by

Avg.TAT = (0 + 24 + 100 + 115 + 93) / 5

= 66.4

Number of Context Switches = 06

Example 2:

TABLE 3: (Example 2 Using Conventional Round Robin Approach)

TIME QUANTUM = 10 ms		
Process Name	**Arrival Time**	**Burst Time**
P1	0	11
P2	0	12
P3	0	13
P4	0	22
P5	0	23
P6	0	33

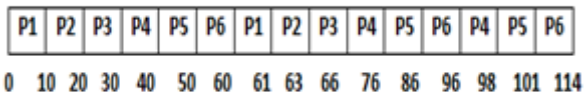

Fig 3: Gantt chart according to classical Round Robin Approach

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for Conventional Round Robin Approach

Average Turn around Time (Avg. TAT) is given by

Avg. TAT = (61 + 63 + 98 + 101 + 114) / 6

= 83.83

Average Waiting Time (Avg. WT) is given by

Avg. WT = (50 + 51 + 76 + 78 + 81) / 6

= 64.83

Number of context Switches = 14

TABLE 4: (Example 2 Using Proposed Approach)

TIME QUANTUM = 10 ms				
Process Name	**Arrival Time**	**Burst Time**	**B = INT(BT / TQ)**	**C = BT % TQ**
P1	0	11	1	1
P2	0	12	1	2
P3	0	13	1	3
P4	0	22	2	2
P5	0	23	2	3
P6	0	33	3	3

Calculating the value of K as follows

K = ceil (Average of (C[i]))

K=ceil (1 +2+3+2+3+3) / 6)

K= ceil (14/6)

K=ceil (2.33)

K=3

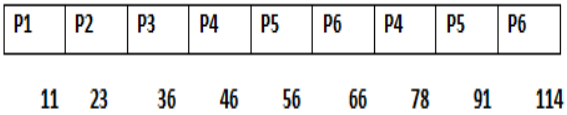

Fig 4: Gantt chart according to proposed algorithm

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for Conventional Round Robin Approach

Average Turn around Time (Avg. TAT) is given by

Avg. TAT = (11+ 23 + 36 + 78 + 91 + 114) /6

= 58.83

Average Waiting Time (Avg. WT) is given by

Avg. WT = (0 + 11+ 23 + 56 + 68 + 81) / 6

= 39.83

Number of Context Switches =08

Example 3:

TABLE 5: (Example 3 Using Conventional Round Robin Approach)

	Time Quantum (TQ) = 10	
Process Name	**Arrival Time**	**Burst Time**
P1	0	11
P2	0	12
P3	0	29
P4	0	30
P5	0	27
P6	0	13

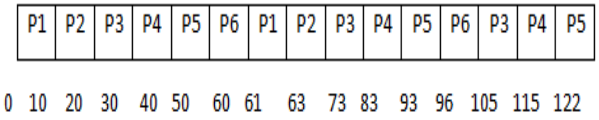

Fig 5: Gantt chart according to Conventional Round Robin Approach

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for Conventional Round Robin Approach

Average Turn around Time (Avg. TAT) is given by

Avg. TAT = (61 + 63 + 105 + 115 + 122 + 96) /6

= 93.66

Average Waiting Time (Avg. WT) is given by

Avg. WT = (50 + 51 + 76 + 75 + 95 + 83) / 6

= 71.66

Number of Context Switches = 14

TABLE: 6 (Example 3 Using proposed Approach)

Time Quantum (TQ) = 10				
Process Name	**Arrival Time**	**Burst Time**	**B = INT(BT / TQ)**	**C = BT % TQ**
P1	0	11	1	1
P2	0	12	1	2
P3	0	29	2	9
P4	0	30	3	0
P5	0	27	2	7
P6	0	13	1	3

Calculating the value of K as follows

K = ceil (Average of (C[i]))

K = ceil (1+2+9+7+3) /5

K =ceil (22/5)

K = ceil (4.4)

K = 5

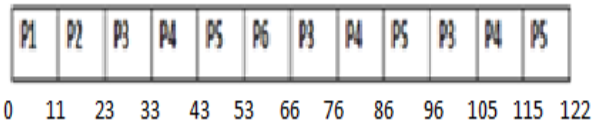

Fig 6: Gantt chart according to proposed algorithm

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for proposed Approach

Average Turn around Time (Avg. TAT) is given by

Avg. TAT = (11+ 23 + 105 + 115 + 122 + 66) / 6

= 72

Average Waiting Time (Avg. WT) is given by

Avg. WT. = (0 +11+ 76+ 75+ 95+ 53) / 6

= 51.66

Number of Context Switches = 11

Example 4:

TABLE 7: (Example 4 Using Conventional Round Robin Approach)

TIME QUANTUM (TQ) = 10 MS		
Process Name	**Arrival Time**	**Burst Time**
P1	0	11
P2	0	12
P3	0	21
P4	0	32
P5	0	23
P6	0	13

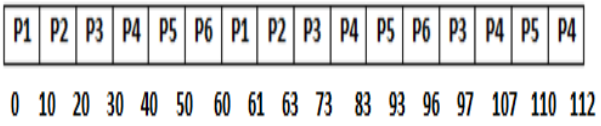

Fig 7: Gantt chart according to conventional round robin policy

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for Conventional Round Robin Approach

Average Turn around Time (Avg. TAT) is given by

Avg. TAT = (61+63+97+112+110+97) / 6

= 88.6

Average Waiting Time (Avg. WT) is given by

Avg. Wait Time = (50+51+76+80+87+83) / 6

= 71.66

Number of Context Switches: 15

TABLE 8: (Example 4 Using Proposed Approach)

TIME QUANTUM (TQ) = 10 MS				
Process Name	**Arrival Time**	**Burst Time**	**B = INT(BT / TQ)**	**C = BT % TQ**
P1	0	11	1	1
P2	0	12	1	2
P3	0	21	2	1
P4	0	32	3	2
P5	0	23	2	3
P6	0	13	3	3

Calculating the value of K as follows

K = ceil (Average of (C[i]))

K = ceil ((1+2+1+2+3+3)/6)

K = ceil (12/6)

K = 2

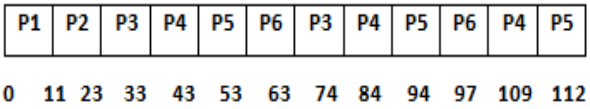

Fig 8: Gantt chart according to the proposed approach

Calculation of Average Turn around Time, Average Waiting Time and Number of Context Switches for proposed Approach

Average Turn around Time (Avg. TAT) is given by

Avg.TAT = (11 + 23 + 74 + 109 + 112 + 97) / 6

= 426 / 6

= 71

Average Waiting Time (Avg. WT) is given by

Avg. WT = (0 + 11 + 53 + 77 + 89 + 84) / 6

= 314 / 6

= 52.33

Number of Context Switches = 11

4. RESULTS AND CONCLUSION

In this paper we have considered round robin algorithm for improvement in the sense of reducing the average turnaround time, average waiting time and the number of context switches. We have presented our algorithm supported by a number of examples with hypothetical data and we have observed a good amount of improvement in scheduling an execution process.

TABLE 9: Comparison Table for Example 1

Attribute for Comparison	**Conventional Round Robin Algorithm**	**Our Proposed Approach**	**Amount of Improvement Observed in our Algorithm**
Average Turn Around Time	129.6	97.4	32.2 units of time saved
Average Waiting Time	98.6	66.4	32.2 units of time saved
Number of Context Switches	09	06	03 number of context switches reduced

TABLE10: Comparison Table for Example 2

Attribute for Comparison	**Conventional Round Robin Algorithm**	**Our Proposed Approach**	**Amount of Improvement Observed in our Algorithm**
Average Turn Around Time	83.83	58.8	25.03 units of time saved
Average Waiting Time	64.83	39.8	25.03 units of time saved
Number of Context Switches	14	08	06 number of context switches reduced

TABLE 11: Comparison Table for Example 3

Attribute for Comparison	**Conventional Round Robin Algorithm**	**Our Proposed Approach**	**Amount of Improvement Observed in our Algorithm**
Average Turn Around Time	93.66	72	21.66 units of time saved
Average Waiting Time	71.66	51.66	20 units of time saved
Number of Context Switches	14	11	03 number of context switches reduced

TABLE 12: Comparison Table for Example 4

Attribute (for Comparison)	**Conventional Round Robin Algorithm**	**Our Proposed Approach**	**Amount of Improvement Observed in our Algorithm**
Average Turn Around Time	88.6	71	17.6 units of time saved
Average Waiting Time	71.66	52.33	19.33 units of time saved
Number of Context Switches	15	11	04 number of context switches reduced

The above tables very evidently depicts that the proposed modification for round robin scheduling technique performs much better than the conventional round robin algorithm.

5. REFERENCES

[1] Samih M. Mostafa, S. Z. Rida, Safwat H. Hamad, "Finding time quantum of round robin CPU scheduling, algorithm in general computing systems using integer programming" October 2010, IJRRAS 5 (1)

[2] Saroj Hiranwal and K. C. Roy, Adaptive Round Robin Scheduling Using Shortest Burst Approach Based On Smart Time Slice, International Journal of Data Engineering (IJDE), Volume 2, Issue 3.

[3] Braunhofer Matthias, Strumflohner Juri, "Fair Round Robin Scheduling", September 17, 2009 URL: matthias-braunhofer-projects.googlecode.com/files /AOE-Report.pdf

[4] H.S.Behera, R. Mohanty, Debashree Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and its Performance Analysis", Volume 5-No. 5, August 2010, International Journal of Computer Application(0975-8887)

[5] Jorge R. Ramos,Veron Rego Janche Sang, "An Improved Computational Algorithm for Round Robin Service" Proceeding of the Winter simulation conference , 2003.

[6] Rami J Matarneh, "Self- Adjustment Time Quantum in Round Robin Algorithm Depending upon the Burts time of the Now Running Processes, American Journal of Applied Sciences 6 (10): 1831-1837, 2009, ISSN 1546-9239.

[7] Haidar M. Ali, Kaies Khalid, "An Improvement on Round Robin Scheduling Method" International Conference on Information Technology and Natural Sciences, ICITNS-2003.

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

[9] http://bcs.wiley.com/he-bcs/Books?action=resource &bcsId=2217& itemId=0471694665&resourceId =5004

[10] http://www.it.uu.se/edu/course/homepage/ oskomp/vt07/lectures/scheduling_algorithms/ handout.pdf

[11] http://www.cis.upenn.edu/~lee/07cis505/Lec/os-schedule-v2.pdf

[12] http://www.scs.stanford.edu/07au-cs140/notes/ l5.pdf

[13] Prof. Rakesh Mohanty, Prof. H. S. Behera, Khusbu Patwari, Manas Ranjan Das, Monisha Dash, Sudhashree "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm" URL : http://www.rimtengg.com/iscet/proceedings/pdfs/advcomp/126.pdf

[14] Shih-Chiang Tsao, Ying-Dar Lin, "Pre-order De®cit Round Robin: a new scheduling algorithm for packet-switched networks, Computer Networks 35 (2001) 287±305, URL: http://140.113.88.160/~ydlin /DRR.pdf

[15] Sanjaya Kumar Panda, Sourav Kumar Bhoi An Effective Round Robin Algorithm using Min-Max Dispersion Measure in International Journal on Computer Science and Engineering (IJCSE) Vol. 4 No. 01 January 2012.