Priority Based Round Robin CPU Scheduling Algorithm

Atul Agarwal 19BCE0436
Priyanshu Gupta 19BCE0648
Mohammad Ayaazuddin 19BCE0638
Manav Nanwani 19BCE0641

***Abstract*—This project tries to overcome the drawbacks of Priority Based CPU Scheduling System and Round Robin CPU Scheduling System by combining the two and maintaining the advantages of both. The project will implement at least 2 algorithms, one considering arrival time and one without it. The objective will be to make time quantum dynamic and reduce the average turn around and waiting time.**

Keywords—Priority, Round-Robin, CPU Scheduling, Algorithm, Dynamic Time Quantum.

I. INTRODUCTION

Modern operating systems depend highly on CPU Scheduling systems to improve multitasking and multiprogramming. There exist multiple CPU Scheduling algorithms such as First Come First Serve (FCFS), Shortest Job First (SJF), Round-Robin, and Priority-based. Out of these RR and Priority-based are the most frequently used and relied on. However, they too have their own drawbacks. By combining these two and using a dynamic time quantum whenever possible we try to reduce turnaround time and waiting time.

A. Round Robin algorithm (RR)

A clock interrupt is generated at periodic intervals. When the interrupt occurs, the currently running process is placed in the ready queue and the next job to execute is selected. This technique is also known as time slicing because each process is given a slice of time before being preempted. The most important challenges facing the (RR) is the amount of the (Quantum - time slice that is given to process before being preempted) where if it was largest of the biggest process, (RR) faces the risk of switching to (FCFS) and if it is very small, then a smaller process will need two rounds at least to complete its execution. As well as the throughput rate may degrade because of the large waiting time and long turnaround time and a large number of context switches. To make (RR) suitable to work with (RTS), it is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time. In most cases, we optimize the average measure in addition to the minimization of the context switch.

Drawbacks

• If the small-time quantum is given the number of context switches will increase which affects the execution time of a process. If the big-time quantum is given it works in the same manner as first come first serve algorithm

• The shortest burst time processes may have to wait for a long time which causes starvation.

• Can't be used for real-time applications that need faster execution.

B. Priority Based CPU Scheduling(PB)

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. The process with the highest priority is to be executed first and so on. Processes with the same priority are executed on a first-come-first-served basis. Priority can be decided based on memory requirements, time requirements, or any other resource requirement.

C. Turnaround time

Turnaround time of a process can be given as the interval from the time of submission of a process to the time of completion.

D. Waiting time

The waiting time can be given as the time a process waits in the ready queue to acquire the processor.

E. Response time

Response time is the interval from the time of submission of the request to the time of the first response produced.

II. OBJECTIVES

A. Implement Algorithm for both cases:

We have two algorithms one each for scheduling systems with and without waiting time. Our main objective will be to implement these algorithms.

B. Minimizing turn around time and Waiting time:

An idealistic CPU Scheduling system will have minimal waiting time. Our objective will be to bring the real waiting time as close to the ideal case as possible.

C. Implementing the concept of Priority:

Evaluating a priority for each process and implementing it in the CPU scheduling algorithm will be another major objective of the project.

III. METHODOLOGY

The team will be implementing two algorithms. One will be for CPU Scheduling without arrival times and one with arrival times taken into consideration. Basically one will be preemptive and one will be non-preemptive.

A. Non-Preemptive CPU Scheduling Algorithm:

The method would be to calculate Time Quantum (TQ), using the average of burst time of all processes. The priority for each process P[i] would then be evaluated using BT[i]%TQ. In the case of the same priority, the process with lower i value will be given a higher priority. All the processes are added to the ready queue on bases of their priority and then evaluated using Round Robin CPU Scheduling using the new time quantum(TQ) calculated. We expect a lower number of context switches and lower average waiting times and lower average turnaround times.

B. Preemptive CPU Scheduling Algorithm:

For a preemptive system, the time quantum would have to be dynamic. It would alter at the arrival of each new process. The time quantum would be calculated as the sum of (all remaining burst times/number of processes). Priority for a process would be calculated using a formula specified as WF = (PR*0.8) + (BT*0.7) + (ATT * 0.2). Where PR is specified priority, BT is burst time for that process and ATT is the arrival time for the process.

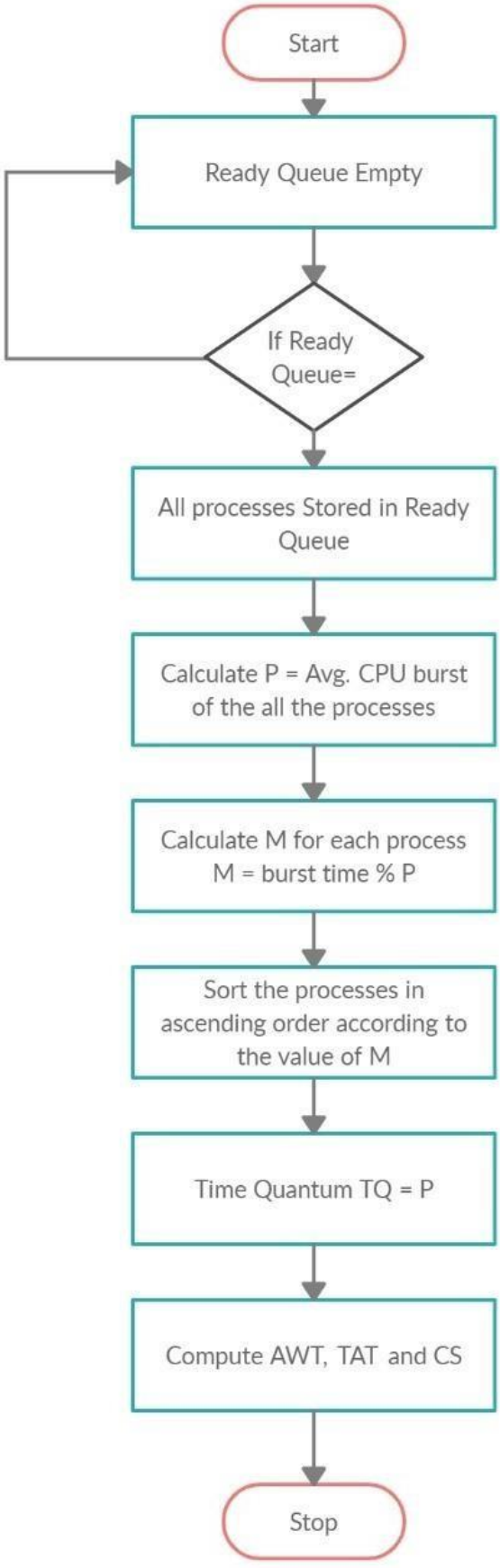
.

IV. ALGORITHM

We will implement two different algorithms in this project. A descriptive analysis of each is given below. One algorithm uses the concept of dynamic time quantum while the other has a fixed time quantum. Both the algorithms evaluate their own time quantums using the Burst Time for each process.

Each algorithm also has its own method of evaluating a priority for each process. One uses Weight Factor while the other uses the remainder of division of burst time for the process and the evaluated time quantum Both the algorithms try to mainly overcome one drawback of the Round Robin CPU Scheduling Algorithm, a small time quantum. However a large time quantum would lead to unutilised rest time, hence priority is utilised.

A. Non- Preemptive Algorithm 1



B. Non Preemptive Algorithm 2

Similar to NON-PREEMPTIVE ALGORITHM 1 but here all the processes are sorted in descending order instead of ascending order. Priority assignment procedure remains same as priority for a process is burst time[i]%P, where P is the average of all the burst times.

Time quantum would be P and all the processes are allotted execution time using modified round robin method, which is explained later.

C. Non Preemptive Algorithm 3

Similar to NON PREEMPTIVE ALGORITHM 1, but here the priority for all the processes would be a product of BT[i] *PRI[i], where BT[i] is the burst time of the process and PRI[i] is an existing input given by the user. All the processes are allotted execution time using modified round robin method, which is explained later.

Everything else remains the same.

D. Non Preemptive 4

Similar to NON PREEMPTIVE ALGORITHM 1, but here the priority for each process would be its own burst time.

All the processes would be sorted according to this priority.

All the processes are allotted execution time using modified round robin method, which is explained later.

E. Preemptive Algorithm 1

We receive data consisting of the Burst Time (BT), Arrival Time (AT), Priority (PR), for each process. A weight factor is calculated for each process (Pi). It is given as:

WTi = PRi*0.8 + BTi*0.7 + ARi*0.2

Using this weight factor we assign new priorities to all the processes. A Dynamic Time Quantum is evaluated, it is given as:

$$TQ = \sum_{i=1}^{n} \frac{remaining\ CPU\ burst}{no.\ of\ process}$$

where n is the number of processes in the ready queue.

The time quantum changes every time a new process arrives. The steps followed in the algorithm are:

1) Start

2) For each process Pi, calculate a Weight Factor WFi, using the formula given above.

3) Arrange processes in decreasing order wrt their WF. When two processes have the same WF consider the priority for arranging.

4) Calculate the Time Quantum, for processes in the ready queue.

5) Assign TQ to process: Pi ->TQ

6) If (i<n) then go to step 5

7) If a new process arrives

Update the counter n and go to step 2

End of iteration

8) Calculate the Average Waiting Time, Average Turnaround Time, Completion Time and the number of context switches.

9) Stop.

F. Preemptive Algorithm 2

Similar to PREEMPTIVE ALGORITHM 1 but here all the processes are sorted in descending order instead of ascending order. Priority assignment procedure remains same as priority for a process is **WTi = PRi*0.8 + BTi*0.7 + ARi*0.2**

, where WT[i] would be calculated priority.

PR[i] is priority input by user, BT[i] is burst time of the process, AR[i] is the arrival time of the process.

Time quantum would be the average of all the burst times of the processes present in the ready queue.

All the processes are allotted execution time using modified round robin method, which is explained later.

G. Preemptive 3

Similar to PREEMPTIVE ALGORITHM 1 but here priority assignment procedure would be done as WT[i]=BT[i]

BT[i] is burst time of the process, WT[i] is Weight Factor for the process.

Time quantum would be the average of all the burst times of the processes present in the ready queue.

All the processes are allotted execution time using modified round robin method, which is explained later.

H. Preemptive 4

Similar to PREEMPTIVE ALGORITHM 1 but here priority assignment procedure would be done as WT[i]=BT[i]*AT[i]

BT[i] is burst time of the process, WT[i] is Weight Factor for the process, AT[i] is the arrival time for the process.

Time quantum would be the average of all the burst times of the processes present in the ready queue.

All the processes are allotted execution time using modified round robin method, which is explained later.

I. MODIFIED ROUND ROBIN CPU SCHEDULING

Here all the process in the ready queue are allotted the time quantum one after another but with a difference.

Incase any process has burst time(x) less than the time quantum(y) then the algorithm will run through the ready queue again to check if other process has a burst time less than or equal to the difference (y-x).

This reduces the average waiting time for most cases.

V. Process Module

A. NON PREEMTIVE:

There are multiple functions in the program. They are listed below:

i. void sort()

ii. void rr()

iii. int main()

First the main module is executed. Here the user enters the number of processes ‘int n’ and accordingly arrays int pid[n], int bt[n], int pri[n] are declared to store the process information. Also a variable int avg is declared. This stores the average of all the burst times and is later used as time quantum.

Next priority is assigned to each process as bt[i]%avg and stored in pri[i].

All the arrays are then passed into the sort() function which will sort them into ascending order according to the priority calculated. In case of same priority the smaller process burst time is taken first.

After sorting all processes enter the rr() (Round Robin) function and every process in the ready queue is assigned time quantum. Here if any process has a burst time less than that of the time quantum then the remaining time is assigned to the remaining time quantum. The Turn Around Time and Average Waiting Time is Calculated and displayed.

B. PREEMTIVE:

This program too has multiple modules and functions. They are listed as:

i. int time_q()

ii. void sort()

iii. void rr()

iv. int main()

First the user enters the number total of processes (int n). The details of all the processes are stored in:

int pid[n],bt[n],at[n],pri[n],tc,flag[n],i;

int wt_fac[n],og_burst[n];

flag[i] is initially 0 for all, and wt_fac is calculated as:

wt_fac[i]=(pri[i]*8)+(bt[i]*7)+(at[i]*2);

The initial time evolved is the arrival time of the first process. All the variables are then transferred to the void rr() function.

In the round robin function all the processes with arrival time less than or equal to the arrival time of the first process are to the ready queue and the variable int num is used to record the number of process in the ready queue. The ready queue variables are:

int i,j,time=time_quantum,bur[no],p[no],art[no],

weight[no],num=0,obur[no],ct[no],tat[no],wt[no];

Next the ready queue int bur[num] is passed to the time_q function to calculate the time quantum. Here the average of all the burst times is calculated and that is the time quantum. All the processes are then sorted according to their existing weight factor that is calculated. Next all the processes are assigned the required time quantum according to the algorithm.

VI. IMPLEMENTATION

A. Non Preemptive:

TEST CASES: NON PREEMPTIVE

PROCESS ID	BURST TIME
P1	23
P2	12
P3	9
P4	10
P5	16

TYPE	1	2	3	4
AVG. TAT	45.599	43.400	48.799	38.00
AVG WT	31.666	29.400	34.799	24.00

TEST CASES: NON PREEMPTIVE

PROCESS ID	BURST TIME
P1	10
P2	8
P3	2
P4	6
P5	9
P6	12

TYPE	1	2	3	4
AVG TAT	30.66	32.00	28.33	29.166
AVG WT	22.833	24.166	20.5	21.333

TEST CASES: NON PREEMPTIVE

PROCESS ID	BURST TIME
P1	20
P2	15
P3	10
P4	5
P5	25

TYPE	1	2	3	4
AVG. TAT	41.00	52.00	48.00	38.00
AVG WT	26.00	37.00	33.00	23.00

TEST CASES: NON PREEMPTIVE

PROCESS ID	BURST TIME
P1	5
P2	3
P3	1
P4	2
P5	3

TYPE	1	2	3	4
AVG. TAT	5.2	7.00	7.4	7.800
AVG WT	8.00	9.800	10.2	5.00

TEST CASES: NON PREEMPTIVE

PROCESS ID	BURST TIME
P1	5
P2	6
P3	7
P4	9
P5	2
P6	3

TYPE	1	2	3	4
AVG TAT	17.66	17.00	17.67	17.166
AVG WT	12.33	22.33	23.00	11.833

TEST CASES: NON PREEMPTIVE

PROCESS ID	BURST TIME
P1	4
P2	5
P3	2
P4	1
P5	6
P6	3

TYPE	1	2	3	4
AVG TAT	12.833	13.66	10.5	10.8333
AVG WT	9.33	10.166	14.00	7.333

B. Preemptive:

TEST CASES: PREEMPTIVE

PROCESS ID	ARRIVAL TIME	BURST TIME	PRIORITY
P1	0	5	1
P2	1	3	2
P3	2	1	3
P4	3	2	4
P5	4	3	5

TYPE	1	2	3	4
AVG TAT	7.600	8.600	7.200	7.600
AVG WT	4.800	5.600	4.400	4.800

TEST CASES: PREEMPTIVE

PROCESS ID	ARRIVAL TIME	BURST TIME	PRIORITY
P1	0	4	1
P2	1	5	2
P3	2	2	3
P4	3	1	4
P5	4	6	5
P6	6	3	6

TYPE	1	2	3	4
AVG TAT	9.500	12.00	9.666	9.33
AVG WT	6.00	8.50	6.166	5.833

TEST CASES: PREEMPTIVE

PROCESS ID	ARRIVAL TIME	BURST TIME	PRIORITY
P1	5	5	1
P2	4	6	2
P3	3	7	3
P4	1	9	4
P5	2	2	5
P6	6	3	6

TYPE	1	2	3	4
AVG TAT	21.833	20.00	22.000	25.500
AVG WT	16.500	22.66	16.66	20.166

TEST CASES: PREEMPTIVE

PROCESS ID	ARRIVAL TIME	BURST TIME	PRIORITY
P1	0	40	1
P2	1	20	2
P3	2	10	3
P4	3	15	4
P5	4	5	5

TYPE	1	2	3	4
AVG TAT	60	72	60	61
AVG WT	42	54	42	42.6

VII. CONCLUSION

For Non Preemptive:

In most of our test cases the best turnaround time and the waiting time were lowest for our algorithm where the processes were sorted in ascending order of their burst times and the time quantum was assumed to be the average of all the burst times given.

Hence we can conclude that the proposed algorithm is better than the existing Non-Preemptive Algorithm.

For Preemptive:

Here too in most cases the algorithm which provided the best turnaround time and waiting time was the one where processes (in ready queue) were prioritised according to the burst time and sorted in increasing order.

Hence we can conclude that the proposed algorithm is better than the existing Preemptive Algorithm.

VIII. FINAL CONCLUSION

After looking at the test cases of both preemptive and non preemptive algorithms that we considered, we conclude that the better algorithm for Round Robin CPU Scheduling is to

i) sort the processes in increasing order on the bases of their burst time

ii) to take a time quantum equal to the average of the burst of all present in the ready queue.

REFERENCE

The team went through multiple research papers and online algorithms to find the ones to implement for the project.

[1] Dash, S. S., Bakhara, N., Agrawalla, P., & Behera, P. P. (2018). A New Proposed Dynamic Quantum for Priority Based Round Robin Scheduling Algorithm. *International Research Journal of Engineering and Technology*, *5*(2).

[2] Joshi, A., & Gosswami, S. (2017). Modified Round Robin algorithm by using Priority Scheduling. *Advances in Computational Sciences and Technology*, *10*(6), 1543-1549.

[3] Rajput, I. S., & Gupta, D. (2012). A priority-based round robin CPU scheduling algorithm for real-time systems. *International Journal of Innovations in Engineering and Technology*, *1*(3), 1-11.

[4] Zouaoui, S., Boussaid, L., & Mtibaa, A. (2019). Priority based round robin (PBRR) CPU scheduling algorithm. *International Journal of Electrical & Computer Engineering (2088-8708)*, *9*(1).

[5] Mohanty, R., Behera, H. S., Patwari, K., Dash, M., & Prasanna, M. L. (2011). Priority-based dynamic round-robin (PBDRR) algorithm with intelligent time slice for soft real-time systems. *arXiv preprint arXiv:1105.1736*.

[6] Katoch, P. C., & Sharma, I. Modified Round Robin Scheduling Algorithm Based on Priorities**.**

[7] Khalil, Z. H., & Alaasam, A. B. A. (2013). Priority-based dynamic round robin with intelligent time slice and highest response ratio next algorithm for soft real-time system. *Global Journal of Advanced Engineering Technologies*, *2*(3), 120-124.

[8] Srivastav, A. P. M., Pandey, S., Gahoi, I., & Namdev, N. K. (2012). Fair priority Round Robin with dynamic time quantum: FPRRDQ. *International Journal of Modern Engineering Research (IJMER)*, *2*, 876-881.

[9] Mohanty, R., Behera, S. R., & Pradhan, S. C. (2012). Apriority-based dynamic round robin with deadline (PBDRRD) scheduling algorithm for hard real-time operating system. *International Journal of Advanced Research in Computer Science*, *3*(3).

[10] Shafi, U., Shah, M. A., Wahid, A., Abbasi, K., Javaid, Q., Asghar, M., & Haider, M. (2020). A novel amended dynamic round-robin scheduling algorithm for time shared systems. *Int. Arab J. Inf. Technol.*, *17*(1), 90-98.

[11] Fataniya, B., & Patel, M. (2018). Dynamic time quantum approach to improve round-robin scheduling algorithm in cloud environment. *IJSRSET*, *4*(4), 963-969.

[12] 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.

[13] Khalil, Z. H., & Alaasam, A. B. A. (2013). Priority-based dynamic round robin with intelligent time slice and highest response ratio next algorithm for soft real-time system. *Global Journal of Advanced Engineering Technologies*, *2*(3), 120-124.

[14] Rushee, K. I., Kalpoma, K. A., & Akter, T. Improvised Priority-based Round Robin CPU Scheduling. *International Journal of Computer Applications*, *975*, 8887.

[15] Singh, A., Goyal, P., & Batra, S. (2010). An optimized round-robin scheduling algorithm for CPU scheduling. *International Journal on Computer Science and Engineering*, *2*(07), 2383-2385.

[16] Samanta, B., & Kumar, B.(2017) A Comparative Study on PBRR CPU Scheduling Algorithm for Different Time Quantum and Priority. *International Journals of Advance Research in Computer Science and Software Engineering*, 2277-128X

[17] Rajput, I. S., & Gupta, D. (2012). A priority based round robin CPU scheduling algorithm for real time systems. *International Journal of Innovations in Engineering and Technology*, *1*(3), 1-11.

[18] Govind Prasad Arya, Kumar Nilay & Devendra Prasad (2018). An Improved Round Robin Cpu Scheduling Algorithm based on Priority of Process. *International Journal of Engineering & Technology* 238- 241

[19] Khatri, J. (2016). An enhanced Round Robin CPU scheduling algorithm. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *18*(4), 20-24.

[20] Gulzar, M., Babu, L. K. R., Babu, A. V., & Kumar, S. U. Self- Regulated Priority based Round Robin Scheduling Algorithm.