Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 165 (2019) 363–369

Procedia Computer Science

www.elsevier.com/locate/procedia

INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ADVANCED COMPUTING 2019, ICRTAC 2019

A Modified Priority Preemptive Algorithm for CPU Scheduling

Kunal Chandiramani*, Rishabh Verma, Sivagami M

Vellore Institute of Technology, Vandalur Kelambakkam Road, Chennai, 600127, India

Abstract

Priority Preemptive scheduling algorithm is a popular among various other algorithms for scheduling CPU, however it leads to the problem of starvation which happens when processes with lower priority are not given any chance of CPU utilization due to continuous CPU usage by processes with higher priorities. To eradicate this problem here, we have proposed a new algorithm for scheduling CPU which we call it as Modified Priority Preemptive Scheduling Algorithm based on a new approach where we are executing priority Preemptive scheduling in a round robin fashion taking the time quanta equal to the shortest burst time of all the available processes. On analyzing the results, it is observed that modified Preemptive algorithm not only solves the problem of starvation but also enhances the performance of regular Priority Preemptive algorithm by lowering the mean turnaround time and mean waiting time.

Keywords: Operating system, Priority Preemptive, Scheduling, Time quanta, Mean turnaround time, Mean waiting time

1. Introduction

Scheduling is a way of allocating threads or processes to the CPU. This task of allocating the process to the CPU is performed by the operating system and these techniques are very important when we have several tasks running at the same time. Scheduling is the most important paradigm to make efficient utilization of the resources. The several parameters that are generally used for deciding the efficiency of any algorithm are given below:

- Making use of CPU: The period of the processes in execution and which keep the CPU busy. Scheduling algorithms aim for increasing CPU utilization. That means, they try to maintain the CPU busy and utilize this resource as much as it can.
- Throughput: It is defined as the number of processes that can be executed per unit time. For better efficiency throughput should be high. [4][5][6]

* *Corresponding author. Tel.: +91-7972366632*
E-mail address: kunal.kc.chandiramani@gmail.com

10.1016/j.procs.2020.01.037

- Waiting time: It is defined as the period between when the process arrives and is allocated to the processor. [4][5][6]
- Response time It is defined as the period between when the processes arrive and till the production of first response. Link with each process the length of the subsequent CPU burst. [4][5][6]
- Turn-around time: The quantity of time spent from it's entry in ready queue till it exits is called the turn-around time. [4][5][6]
- Operating Systems: Operating system behaves like the bridge connecting the user to the hardware. It is the core of the computer system which executes the application software from the client-side. It bridges the gap between the computer system and humans and acts as an intermediary.
- CPU Scheduling: when there are two or more processes in the ready state they cannot be executed at the same time in a single-core CPU and therefore needs to be scheduled so that each process can be executed one by one and can also be executed with appropriate priority.

This literature consists of four more sections, the second fragment is about literature Review, third is about the proposed algorithm, fourth is Experimental Results and Analysis and finally, the fifth segment is conclusion and discussion.

2. Literature Survey

In this segment of the paper, we have elucidated about different kinds of literature that are related to Priority Preemptive CPU scheduling algorithm. Multiple papers are published in this domain of computer science dealing with the operating system and its scheduling algorithms in recent years,some of them are discussed below,

Rakesh Patel et al [1] found that an alternate to calculate time unit or quanta optimally by taking the shortest processing time for the set of all the processes of the queue in the round robin algorithm while considering that the arrival time of all the processes is 0.

Muhammad Akhtar et al [2] proposed a new technique for calculating the time quanta, as an attempt to minimize the context switching and mean Turnaround time by introducing that if the burst time of the process is greater than half of the remaining execution time, then running process will not contact switched. H.S. Behera et al [3] have used a mathematical approach for the calculation of time quanta using standard deviation and mean.

SARR algorithm [8] proposed a new method known as dynamic time-quantum, here the factor is adjusted frequently in accordance of the processing time of the process running in the CPU. Mixed Scheduling [9], proposed an approach where it mixes the sequence of the processes of the popular scheduling algorithms which are non-Preemptive, first come-first serve and Shortest job first. In this algorithm, among all processes ready to be executed, process that has least execution period is completed first and after this, the next least from the list and so on. Burst Round Robin (BRR) [10], new weighting method is implemented. Here processes with shorter jobs are given relatively higher priority, this is done to reduce the waiting time of the shorter jobs.

Harsha Parekh et al [12] in their paper have proposed a method in which they have categorized the priority of the tasks into three different categories which are low priority processes executing for 80% of the fixed time quanta, high priority processes executing for 120% of the fixed time quanta while the time quanta of the medium priority process remain the same.

In 2015, Amar Ranjan et al [13] proposed DABRR Algorithm where if the remaining burst time is less than the time quanta then it is executed completely else it is executed for the period of the time quanta to reduce context switching. An improvement to [13] in 2018 Hina Gull et al[14] proposed an algorithm where all of the processes present in the queue are divided into two equal parts and the lists values will be calculated separately based on its average. The two separate values and the sum of these two will be the time quantum for each process.

3. Proposed Work

Simple Priority Preemptive scheduling algorithm works by executing the process which has the highest priority first, even when a new process with higher priority arrives it stops the execution of the currently running process and gives chance to the newly arrived higher priority process. However, it undergoes a severe problem of high waiting

time for low priority processes. This happens when higher priority processes with large burst times are executed first and continues CPU utilization instead of giving chance to processes with lower priority even after having low burst time.

To eradicate this problem in this paper, we have proposed a scheduling algorithm which works based on a new approach where we are executing priority Preemptive scheduling in a round robin fashion taking the time quanta equal to the shortest burst time of all the available processes. In addition to that, it also tries to reduce the mean waiting time, mean response time and mean turnaround time of simple priority Preemptive scheduling by giving the chance to low priority jobs in round robin order. The main objective of this paper is to analyze and propose a CPU Scheduling algorithm to mitigate the mean turnaround time, mean response time and mean waiting time. The detailed explanation of the proposed algorithm is discussed in section 3.1.

3.1. Pseudocode : Modified Priority Preemptive (MPP) CPU scheduling

1. Initialize all the processes by allocating Execution Time, Priority and Arrival Time to each process.
2. compare the burst time (execution time) of all the processes and store the shortest burst time as a new variable which will be equal to time quanta of the algorithm.
3. Store all the processes in a data structure and sort these processes based on their priorities.
4. Execute these processes in round robin fashion one by one based on their priority with higher priority process executing first and with time quanta equal to the shortest burst time.
5. Check if the remaining burst time of the currently executing process is less than or equal to the half of the time quanta, if yes than execute the process completely else continue to next process.
6. Steps 4 and 5 are repeated until all the process have finished execution.

4. Experimental Results and Analysis

The proposed algorithm has been implemented in C in Windows 10.0 platform. The proposed algorithm has been evaluated using the metrics mean turnaround time and mean waiting time and to analyse the performance of the proposed algorithm, it has been compared with simple priority Preemptive scheduling algorithm[7] and priority based round robin scheduling algorithm[11].

To check the effectiveness of the proposed algorithm, we have considered five different cases to compare modified priority Preemptive scheduling algorithm with simple priority Preemptive scheduling algorithm[7] and priority based round robin CPU scheduling algorithm[11]. We have considered 5 different test cases, Case 1 with 3 processes, Case 2 with 5 processes, Case 3 with 4 processes, Case 4 with a large dataset of 25 processes and finally case 5 whose dataset is same as in Ref [11].

It has been found that the proposed algorithm gives decreased turnaround time and waiting time in comparison to both, simple Priority Preemptive scheduling algorithm[7] and priority based round robin CPU scheduling algorithm[11]. The comparison results of the proposed work with simple priority Preemptive scheduling algorithms and priority based round robin CPU scheduling algorithm [11] has been shown in Table 1,2,3,4,5 and in Figure 1,2,3,4,5.

4.1. Case 1

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
0	0	2	4	17	17	15	15
1	0	5	6	15	15	10	10
2	0	10	10	10	10	0	0

MEAN WAITING TIME:8.333333

MEAN TURNAROUND TIME:14.000000 MEAN RESPONSE TIME:8.333333

(a) Simple Priority Preemptive.

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
2	0	10	10	17	17	7	0
1	0	5	6	11	11	6	2
0	0	2	4	6	6	4	4

TIME QUANTA:2 MEAN WAITING TIME:5.666667

MEAN TURNAROUND TIME:11.333333 MEAN RESPONSE TIME:2.000000

(b) Modified Priority Preemptive.

Fig. 1: Comparison between Simple and Modified Priority Preemptive Algorithm.

Table 1: Performance Comparison for case 1.

Algorithm	Mean turnaround time	Mean waiting time	Mean Response time
Simple Priority Preemptive	14.00	8.33	8.33
Proposed Priority Preemptive	11.33	5.66	2.00

4.2. Case 2

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
0	0	5	7	20	20	15	15
1	0	2	6	22	22	20	20
2	0	3	4	25	25	22	22
3	0	4	3	29	29	25	25
4	0	15	10	15	15	0	0

MEAN WAITING TIME:16.400000

MEAN TURNAROUND TIME:22.200001 MEAN RESPONSE TIME:16.400000

(a) Simple Priority Preemptive.

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
4	0	15	10	29	29	14	0
0	0	5	7	16	16	11	2
1	0	2	6	6	6	4	4
2	0	3	4	9	9	6	6
3	0	4	3	18	18	14	9

TIME QUANTA:2 MEAN WAITING TIME:9.800000

MEAN TURNAROUND TIME:15.600000 MEAN RESPONSE TIME:4.200000

(b) Modified Priority Preemptive.

Fig. 2: Comparison between Simple and Modified Priority Preemptive Algorithm.

Table 2: Performance Comparison for case 2.

Algorithm	Mean turnaround time	Mean waiting time	Mean Response time
Simple Priority Preemptive	22.20	16.40	16.40
Proposed Priority Preemptive	15.60	9.80	4.20

4.3. Case 3

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
0	0	5	4	42	42	37	37
1	0	7	6	37	37	30	30
2	0	12	7	30	30	18	18
3	0	18	10	18	18	0	0

MEAN WAITING TIME:21.250000

MEAN TURNAROUND TIME:31.750000 MEAN RESPONSE TIME:21.250000

(a) Simple Priority Preemptive.

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
3	0	18	10	42	42	24	0
2	0	12	7	34	34	22	5
1	0	7	6	17	17	10	10
0	0	5	4	22	22	17	17

TIME QUANTA:5 MEAN WAITING TIME:18.250000

MEAN TURNAROUND TIME:28.750000 MEAN RESPONSE TIME:8.000000

(b) Modified Priority Preemptive.

Fig. 3: Comparison between Simple and Modified Priority Preemptive Algorithm.

Table 3: Performance Comparison for case 3.

Algorithm	Mean turnaround time	Mean waiting time	Mean response time
Simple Priority Preemptive	31.75	21.25	21.25
Proposed Priority Preemptive	28.75	18.25	8.00

4.4. Case 4

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
0	0	3	3	348	348	345	345
1	0	4	4	345	345	341	341
2	0	18	22	88	88	70	70
3	0	6	2	354	354	348	348
4	0	8	7	331	331	323	323
5	0	5	6	336	336	331	331
6	0	20	10	283	283	263	263
7	0	40	21	128	128	88	88
8	0	32	14	221	221	189	189
9	0	10	9	293	293	283	283
10	0	13	13	234	234	221	221
11	0	21	8	323	323	302	302
12	0	16	11	263	263	247	247
13	0	14	20	142	142	128	128
14	0	3	25	67	67	64	64
15	0	5	20	147	147	142	142
16	0	43	26	64	64	21	21
17	0	28	18	175	175	147	147
18	0	13	13	247	247	234	234
19	0	9	9	302	302	293	293
20	0	14	17	189	189	175	175
21	0	3	25	70	70	67	67
22	0	9	30	21	21	12	12
23	0	12	35	12	12	0	0
24	0	5	5	341	341	336	336

MEAN WAITING TIME:198.800003
MEAN TURNAROUND TIME:212.960007 MEAN RESPONSE TIME:198.800003

PID	AT	BT	PRIORITY	CT	TAT	WT	RT
23	0	12	35	190	190	178	0
22	0	9	30	142	142	133	3
15	0	5	30	84	84	79	6
16	0	43	26	354	354	311	9
14	0	3	25	15	15	12	12
21	0	3	25	18	18	15	15
2	0	18	22	263	263	245	18
7	0	40	21	350	350	310	21
13	0	14	20	239	239	225	24
17	0	28	18	317	317	289	27
20	0	14	17	244	244	230	30
8	0	32	14	337	337	305	33
10	0	13	13	215	215	202	36
18	0	13	13	219	219	206	39
12	0	16	11	251	251	235	42
6	0	20	10	292	292	272	45
9	0	10	9	179	179	169	48
19	0	9	9	182	182	173	51
11	0	21	8	295	295	274	54
4	0	8	7	187	187	179	57
5	0	5	6	131	131	126	60
24	0	5	5	133	133	128	63
1	0	4	4	70	70	66	66
0	0	3	3	73	73	70	70
3	0	6	2	136	136	130	73

TIME QUANTA:3 MEAN WAITING TIME:182.479996
MEAN TURNAROUND TIME:196.639999 MEAN RESPONSE TIME:36.080002

(a) Simple Priority Preemptive. (b) Modified Priority Preemptive.

Fig. 4: Comparison between Simple and Modified Priority Preemptive Algorithm.

Table 4: Performance Comparison for case 4.

Algorithm	Mean turnaround time	Mean waiting time	Mean Response time
Simple Priority Preemptive	212.80	198.80	198.80
Proposed Priority Preemptive	196.63	182.47	36.08

4.5. Case 5

```
Process        Burst Time       Priority
---------      ----------       ---------
   A              22               4
   B              18               2
   C               9               1
   D              10               3
   E               4               5
Enter the value of Time Quantum:9

Processes are executed in the following sequence:
process C for  9 ms
process B for  9 ms
process D for  9 ms
process A for  9 ms
process E for  4 ms
process D for  1 ms
process B for  9 ms
process A for 13 ms

GANTT Chart for the Priority based Round Robin scheduling is :
C---------B---------D---------A---------E----D-B---------A-----

TOTAL CONTEXT SWITCHES:7
AVERAGE WAITING TIME:28.000000 ms
AVERAGE TURNAROUND TIME:40.600000 ms
```

```
Enter process id:1
Enter Arrival Time, burst time and priority:0 22 4

Enter process id:2
Enter Arrival Time, burst time and priority:0 18 2

Enter process id:3
Enter Arrival Time, burst time and priority:0 9 1

Enter process id:4
Enter Arrival Time, burst time and priority:0 10 3

Enter process id:5
Enter Arrival Time, burst time and priority:0 4 5
TIME QUANTA:4
PID     AT      BT      PRIORITY        CT      TAT     WT
5       0       4       5               4       4       0
1       0       22      4               63      63      41
4       0       10      3               30      30      20
2       0       18      2               57      57      39
3       0       9       1               39      39      30
AVERAGE TURNAROUND TIME:38.599998
AVERAGE WAITING TIME:26.000000
```

(a) Priority based round robin[11]. (b) Modified Priority Preemptive.

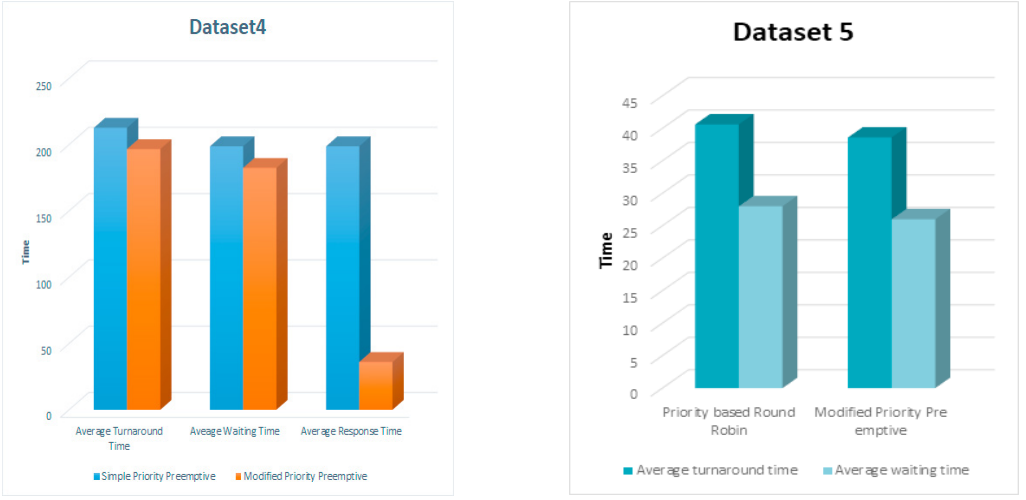
Fig. 5: Comparison between Priority based round robin and Modified Priority Preemptive Algorithm.

Table 5: Performance Comparison for case 5.

Algorithm	Mean turnaround time	Mean waiting time
Priority Based round robin[11]	40.60	28.00
Proposed Priority Preemptive	38.5	26.00

(a) Mean Turnaround Time. (b) Mean Waiting Time. (c) Mean Response Time.

Fig. 6: Comparison of Mean turnaround time, Mean waiting time and Mean response time for Case 1, 2 and 3.



(a) Simple v/s Modified Priority Preemptive.. (b) Priority based round robin v/s Modified Priority.

Fig. 7: Comparison of Mean turnaround time, Mean waiting time and Mean response time for Case 4 and 5.

It can be inferred from the graph of Figure.6(a) that the proposed modified priority preemptive algorithm has decreased Mean turnaround time.In addition to that, the proposed algorithm also has lessen mean waiting time and mean response time than simple priority Preemptive algorithm[7] as shown in graph of Figure.6(b) and Figure.6(c). Furthermore the proposed algorithm also proves its effectiveness by performing better than priority based round robin algorithm[11] by having reduced mean turnaround time, mean response time and mean waiting time as shown in Figure.7(b).

Table 6: Execution time comparison for case 1, 2, 3 and 4.

Case	Simple Priority Preemptive(seconds)	Modified Priority Preemptive(seconds)
Case-1	4×10^{-6}	3×10^{-6}
Case-2	5×10^{-6}	4×10^{-6}
Case-3	4×10^{-6}	3.4×10^{-6}
Case-4	49×10^{-6}	11×10^{-6}

Even though the time complexity of both simple priority preemptive algorithm and proposed algorithm is $O(n^2)$, it can be inferred from Table 6 that the proposed algorithm takes comparatively less time to execute than simple priority preemptive algorithm.

5. Conclusion

The paper discusses a new algorithm for CPU scheduling called Modified Priority Preemptive (MPP) CPU scheduling algorithm which is intended for solving one of the major problems of starvation in simple priority Preemptive algorithm[7] by amalgamating simple priority preemptive with round robin scheduling algorithm, and executing priority Preemptive scheduling in a round robin fashion taking the time quanta equal to the shortest burst time of all the available processes. To check the effectiveness of the proposed algorithm, it has been compared to traditional priority Preemptive algorithm by considering four different cases, and not only the proposed algorithm solves the problemof starvation but also it overwhelms the traditional priority Preemptive algorithm by decreasing mean turnaround time, mean waiting time and mean response time. In addition to that, the proposed algorithm has also been compared to priority based round robin CPU scheduling algorithm[11] in case 5 of section four and the proposed algorithm performs better by reducing the mean turnaround time and mean waiting time. These outcomes of the comparison for different cases demonstrate that the proposed algorithm improves the performance of traditional simple priority Preemptive[7] and priority based round robin CPU scheduling algorithm[11]. Furthermore, the proposed work can be extended for multi-core processors in future.

References

[1] Patel, Rakesh, and M. Patel. "Sjrr cpu scheduling algorithm." Int J Eng Comput Sci 2 (2013): 3396-3399.
[2] Hamayun, Maryam, and Hira Khurshid. "An optimized shortest job first scheduling algorithm for CPU scheduling." J. Appl. Environ. Biol. Sci 5.12 (2015): 42-46.
[3] BEHERA, HIMANSHU SEKHAR, Sreelipa Curtis, and BIAYALAXMI PANDA. "Enhancing the CPU Performance Using a Modified Mean-Deviation Round Robin Scheduling Algorithm for Real Time Systems." journal of global research in Computer science 3.3 (2012): 9-16.
[4] Arpaci-Dusseau, Remzi H., and Andrea C. Arpaci-Dusseau. Operating systems: Three easy pieces. Arpaci-Dusseau Books LLC, 2018.
[5] Silberschatz, Abraham, P. B. Galvin, and Greg Gagne. "Threads." Operating System Concepts (2012): 163-202.
[6] Elmasri, Ramez, Alan Carrick, and David Levine. Operating systems: a spiral approach. McGraw-Hill, Inc., 2009.
[7] Tanenbaum, Andrew S., and Herbert Bos. Modern operating systems. Pearson, 2015.
[8] Matarneh, Rami J. "Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes." American Journal of Applied Sciences 6.10 (2009): 1831.
[9] Mohan, Sunita. "Mixed Scheduling (A New Scheduling Policy)." Proceedings of Insight 9 (2009): 25-26.
[10] Helmy, Tarek, and Abdelkader Dekdouk. "Burst round robin as a proportional-share scheduling algorithm." IEEEGCC 2007 (2007).
[11] Rajput, Ishwari Singh, and Deepa Gupta. "A priority based round robin CPU scheduling algorithm for real time systems." International Journal of Innovations in Engineering and Technology 1.3 (2012): 1-11.
[12] Harshal Bharatkumar Parekh and Sheetal Chaudhari. "Improved Round Robin CPU Scheduling Algorithm. "International Conference on Global Trends in Signal Processing, Information Computing and Communication."(2016).
[13] Amar Ranjan Dash, Sandipta kumar Sahu and Sanjay Kumar Samantra. "AN OPTIMIZED ROUND ROBIN CPU SCHEDULING ALGORITHM WITH DYNAMIC TIME QUANTUM." International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol. 5, No. 1, February 2015.
[14] Hina Gull, Sardar Zafar Iqbal, Saqib Saeed, Mohammad Abdulrahman Alqatani, and Yasser Bamarouf. "Design and Evaluation of CPU Scheduling Algorithms Based on Relative Time Quantum: Variations of Round Robin Algorithm." Journal of Computational and Theoretical Nanoscience Vol. 15, 24832488, 2018.