2012 4th International Conference on Computer Modeling and Simulation (ICCMS 2012)

IPCSIT vol.22 (2012) © (2012) IACSIT Press, Singapore

Design and Performance Evaluation of Precedence Scheduling Algorithm with Intelligent Service Time (PSIST)

Himanshi Saxena[1], Prashant Agarwal[2]

Department of Computer Science and Engineering

Madan Mohan Malviya Engineering College, Gorakhpur Uttar Pradesh, India

[1]himanshi57@gmail.com, [2]pidge.1000@gmail.com

Abstract. .Problem Statement: Extending the concept of Round Robin algorithm (RR) that incorporates priority and suggesting a novel approach that minimizes context switching overhead, average waiting time and turnaround time. Approach: We calculate factor of precedence 'FP' for each process which determines the order of execution of processes, intelligent service time 'IST' for each process which determines time of execution of process in a single round and execute the processes in RR fashion using the calculated parameters. Results: Based on experiments and calculations, the proposed algorithm is successful in reducing afore mentioned problems. Conclusion: Our proposed algorithm can be effective in priority based systems where burst time can be easily predicted.

Keywords. CPU scheduling algorithm, Context switch, Waiting time, Turnaround time, Priority, Round Robin, Time quantum.

1. Introduction

The goal of any CPU scheduling algorithm is to maximise throughput and CPU utilization and minimize turnaround time, average waiting time and response time. Apart from these parameters there are other factors to consider like number of context switches [1] [7] [8].

Existing scheduling techniques based on priority of the process are not fair and suffer from the problem of starvation. If we use RR algorithm then we can achieve fairness but at the cost of neglecting the effect of priority totally and large number of context switches. If we use priority based RR algorithm [3] [5] [9] we still have the problem of large number of context switches. This motivated us to design a new scheduling algorithm that adopts an approach that is fairer, gives consideration to priority of the process and minimizes the number of context switches, turnaround time and average waiting time. We incorporate the concept of IST rather than time quantum; and FP, rather than priority of process to decide the order of execution of processes in a single round.

The rest of the paper is divided as follows: Section II deals with existing algorithms. Section III describes related work. Section IV gives the details of our proposed scheduling algorithm. Section V compares the performance of our proposed scheduling algorithm with Priority based Dynamic Round Robin (PBDRR) [9]. Conclusion and direction for future work is given in section VI.

2. Existing Algorithms

We discuss Priority based algorithm and RR algorithm. Priority based scheduling algorithm assigns CPU to competing processes according to the priority of the process. Priority can be determined by the user or by the system. Priority scheduling algorithm suffers from problem of starvation [1] [8].

In RR algorithm a small unit of time called time quantum is defined. All run able processes are kept in circular queue. The CPU scheduler goes around this queue allocating the CPU to each process for a time interval of one quantum. It does not suffer from problem of starvation and is fair. RR algorithm, however does not take into account the priority of process at all. It also has more context switching overhead than other algorithms [1] [7] [8].

3. RELATED WORK

Efforts have been made to modify RR in order to give better turnaround time, average waiting time and minimize context switches. Changes to time quantum according to the nature of process have been suggested in [2] [3] [4] [5] [6] [9]. Employing the concept of priority along with RR has been suggested in [3] [5] [9]. The concept of dynamic time quantum has been suggested in [4] [6] [9].

4. PROPOSED ALGORITHM

For achieving low average waiting time and turnaround time, we should execute processes with small burst time early. We schedule the processes that arrive at same time. A weight of 0.6 to priority and a weight of 0.4 to burst time are assigned. This ensures that priority of the process is given more consideration than burst time. Note that a higher priority process gets a higher number and shorter process also gets a higher number in our numbering scheme. We now calculate factor of precedence 'FP' which is:

$$FP = 0.6*P + 0.4*BT \quad (1)$$

P is priority and BT is priority number assigned according to shorter burst time. We round off the value of 'FP' to the nearest integer for rest of the calculations.

We propose to give more execution time in a single round to process with high value of FP. A time quantum 'q' is selected taking into account the same considerations that are taken while selecting time quantum for RR algorithm. Now we decide the intelligent service time 'IST' for each process *i*. It is given by:

$$IST_i = FP_i * q \quad (2)$$

We place all the processes in a priority queue. After calculating IST of each process we assign CPU to process with highest value of IST. In case of conflict the process with shorter burst time is given preference. The process decided executes for a period that is equal to IST or its burst time whichever is smallest. In case the set of process with same value of IST finish execution after a single round, we redistribute the value of 'IST' removing the value of IST of finished process and redistributing the value of IST accordingly. For e.g. suppose there are five processes P1, P2, P3, P4, P5. Value of IST for the processes is 12, 16, 8, 8 and 16 respectively. Suppose P1 finishes execution. Then the value of IST for P2 and P5 will remain 16 but for P3 and P4, it will change to 12 and 12 respectively.

Pseudo code of algorithm

```
process_priority_queue = empty
while (true) {
      for each new process {
            add process to process_priority_queue along with its priority P and continue until all process arrive }
      scheduler( );
      if (process_terminate) {
               if (process_priority_queue is empty)
                     wait until new process arrives
                else
                     recalculate_IST ( );  } }
pre_scheduler(q) {
      for each process i
        { Assign BT to all the processes such that process with shorter burst time get higher number
                                          FP_i = 0.6*P + 0.4*BT_i        // Round off FP to nearest integer
                                          IST_i = FP_i * q               // calculate IST  } }
scheduler ( ) {
    pre_scheduler (time_quantum);
    assign CPU to process with highest value of IST.                     // process selector
       if more than one process then assign CPU to process with shorter burst time among them
```

```
                if (burst time < intelligent service time)
                        execute process till its burst time
                 else
                        execute process till IST_i
    select  next process from process_priority_queue }                    // next process
recalculate_IST ( ) {
     if (process_terminate)
        { terminated_IST  ← IST of terminated process
                if (all the processes having same value of  IST as terminated_IST have terminated after a single round)
                  { redistribute IST such that process with smaller value of IST than terminated_IST are incremented
                     to   terminated_IST and rest of the IST are also  changed  in similar way }
                else
                     do nothing } }
```

5. Experiments and Illustration

We compare our algorithm with PBDRR [9] in terms of number of context switches, turnaround time and average waiting time.

5.1. Assumptions

All experiments are assumed to be performed in uniprocessor environment and all the processes are independent from each other. Attributes like burst time and priority are known prior to submission of process. All processes are CPU bound. No process is I/O bound. Processes with same arrival time are scheduled.

5.2.Illustration and Simulation

Case 1) Suppose there are five processes. Burst time and arrival time have been assumed to be in milliseconds. We calculate 'FP' and 'IST' for each process assuming time quantum to be 4 milliseconds. We assign value of BT in such a way that shorter process gets higher number:

TABLE I

Process	Burst time	Priority 'P'	BT	P	'FP'	'IST' for 1st round
P1	5	4	5	4	4.4	4*4 = 16
P2	12	3	4	3	3.4	3*4 = 12
P3	16	5	3	5	4.2	4*4 = 16
P4	21	2	2	2	2.0	2*4 = 8
P5	23	1	1	1	1.0	1*4 = 4

Since P1 and P3 have same 'IST' i.e. 16, we begin with P1 as it is shorter process. Similarly rest of the process execute for 1st round. After 1st round of execution P1, P2 and P3 have finished executing. So we redistribute 'IST' and now 'IST' of P4 becomes 16 and 'IST' of P5 becomes 12. In 2nd round P4 finishes execution. So in 3rd round P5 gets 'IST' as 16. Execution of PSIST and PBDRR is shown:

TABLE II: PSIST

Process	1st round	2nd round	3rd round
P1	16	0	0
P2	12	0	0
P3	16	0	0
P4	8	16	0
P5	4	12	16

TABLE III: PBDRR

Process	SC	1st	2nd	3rd	4th	5th
P1	0	5	0	0	0	0
P2	0	2	3	7	0	0
P3	0	3	5	8	0	0
P4	0	2	3	5	8	3
P5	0	2	3	5	8	5

Gantt chart for both the algorithms is shown:

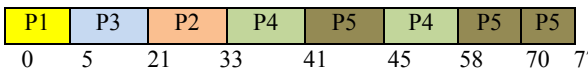

Fig1. Gantt chart with PSIST for case 1

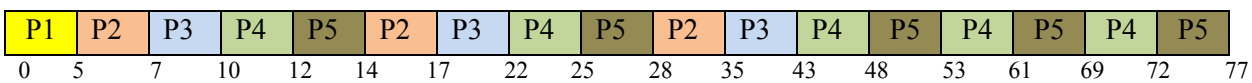

Fig2. Gantt chart with PBDRR for case 1

To calculate the average waiting time, turnaround time and number of context switches we have performed simulations in C on gcc compiler. The snapshot of results of simulation has been shown. We have shown the comparison of several parameters for PSIST and PBDRR in table and through a chart.

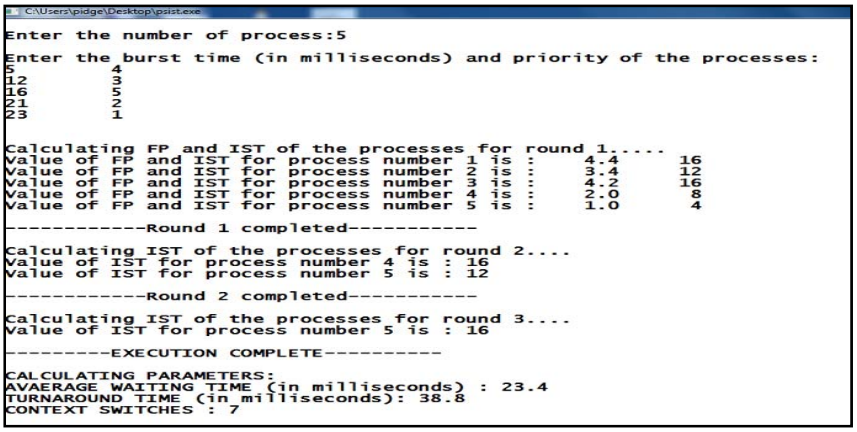

Fig3. Snapshot of simulation of PSIST for case 1

TABLE: IV

Algorithm	Turnaround time (TAT)	Average waiting time (AWT)	Context Switches (CS)
PBDRR	46.4	31.0	17
PSIST	38.8	23.4	7

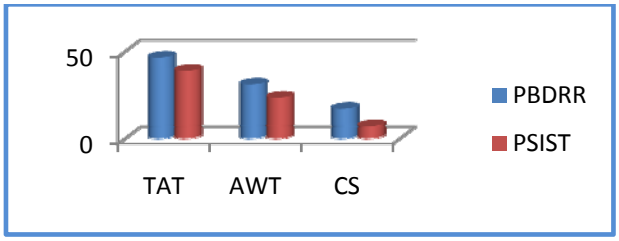

Fig4.Comparison of PBDRR and PSIST for case1

Case 2) Suppose there are five processes. We assume time quantum to be 4 milliseconds:

TABLE V

Process	Burst time	Priority 'P'	BT	P	'FP'	'IST'
P1	11	3	4	3	3.4	3*4 = 12
P2	53	5	1	5	3.4	3*4 = 12
P3	8	4	5	4	4.4	4*4 = 16
P4	41	2	2	2	2.0	2*4 = 8
P5	20	1	3	1	1.8	2*4 = 8

Calculations for PSIST and PBDRR and comparison along with snapshot for the case are shown:

TABLE VI: PSIST

Process	ITS			
	1st round	2nd round	3rd round	4th round
P1	12	0	0	0
P2	12	16	16	16
P3	16	0	0	0
P4	8	12	12	12
P5	8	12	0	0

TABLE VII: PBDRR

Process	SC	Round					
		1st	2nd	3rd	4th	5th	6th
P1	0	2	3	6	0	0	0
P2	0	3	5	8	12	18	7
P3	1	8	0	0	0	0	0
P4	0	2	3	5	8	12	11
P5	1	5	10	5	0	0	0

Gantt chart for both the cases is shown:

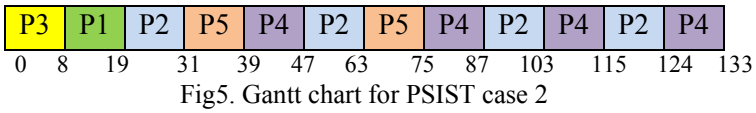

Fig5. Gantt chart for PSIST case 2

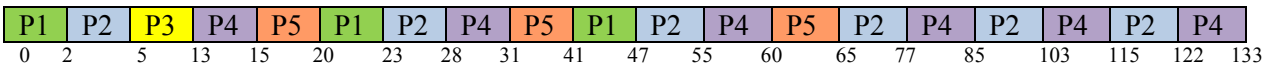

Fig6. Gantt chart for PBDRR case 2

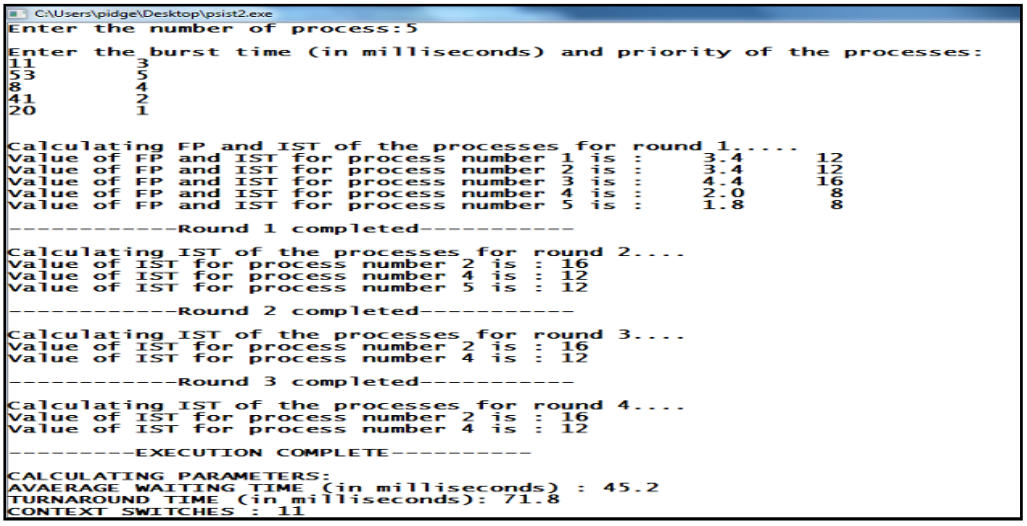

Fig7. Snapshot of Simulation of PSIST for case 2

TABLE VIII: Comparison

Algorithm	Turnaround time (TAT)	Average waiting time (AWT)	Context Switches (CS)
PBDRR	76.0	49.4	18
PSIST	71.8	45.2	11

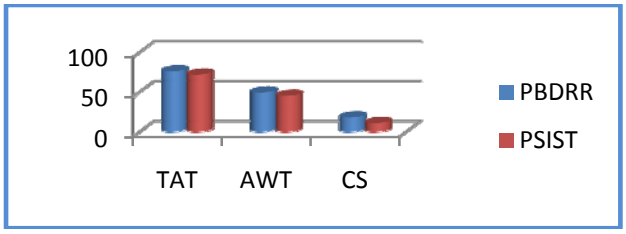

Fig8. Comparison of PBDRR and PSIST on various parameters for case 2

6. Conclusion

From the experimental results we found that our proposed algorithm performs better than PBDRR in terms of decreasing the number of context switches, turnaround time and average waiting time. This has been possible by prefering process with shorter burst time in addition to their priority. Future work can be based on this algorithm that includes process with different arrival times.

7. References

[1]William Stallings, "Operating Systems: Internals and Design Principles" 6th edition, Prentice Hall, ISBN-13:978-0136006329

[2]C. Yaashuwanth and R.Ramesh "A New Scheduling Algorithm for Real Time System", *International Journal of Computer and Electrical Engineering (IJCEE),* Vol.2, No.6, pp 1104 -1106, December (2010)

[3]H.S. Behera, "Comparative and Performance Analysis of Multi-Dynamic Time Quantum Round Robin (MDTQRR) Algorithm with Arrival Time" Indian Journal of Computer Science and Engineering (IJCSE), Vol.2 No.2 Apr-May 2011, ISSN: 0976 – 5166

[4] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm depending on Burst Time of Running Process", *American Journal of Applied Sciences,* ISSN 1546-9239, 6 (10):1831-1837, 2009

[5]Moonju Park, Hong Jin Yoo, Jinseok Chae, "Quantum Based Fixed Priority Scheduling", International *Conference on Advanced Computer Theory and Engineering (2008)*

[6]H.S. Behera "A new Dynamic Round Robin and SRTN Algorithm with Variable Original Time Slice for Soft Real Time Systems", *International Journal of Computer Applications 16 (1):54-60, February 2011*

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

[8] Tanenbaum, Andrew S., "Modern Operating Systems", 3rd edition, Prentice Hall, ISBN: 13:9780136006633

[9] Rakesh Mohanty and H.S. Behera "Priority based Dynamic Round Robin with Intelligent Time Slice for Soft Real Time Systems " , *International Journal of Advanced Computer Science and Applications, Vol.2, No.2, pp. 46- 50*, February 2011