Design and Performance Evaluation of Precedence Scheduling algorithm with Mean Average as Time Quantum (PSMTQ)

P.SURENDRA VARMA

Abstract: Round Robin, considered as most widely used CPU scheduling algorithm depends on the time quantum. If the time quantum chosen is too large, the response time of processes is considered too high. On the other hand, if the time quantum is too small, it increases the overhead of the CPU. The performance of the RR can be improved by taking mean average of burst times as time quantum. The precedence scheduling algorithm is the extension of round robin algorithm which incorporates priority and calculates factor of precedence FP for each process which determines order of execution of processes. In this paper, a novel scheduling algorithm is used which uses mean average as a time quantum and uses the Balanced factor of precedence rather than factor of precedence to find the order of execution of processes. The experimental analysis shows that PSMTQ is better than RR and its variants in terms of reducing the number of context switches, average waiting time and average turnaround time.

Index Terms: Operating System, Scheduling Algorithm, Round Robin, Context switch, Waiting time, Turnaround time.

I. Introduction

CPU scheduling is an essential operating system task, which is the process of allocating the CPU to a specific process for a time slice.

Scheduling requires careful attention to ensure fairness and avoid process starvation in the CPU. Main goal of the scheduling is to maximize CPU utilization, Throughput and to minimize response time, context switches, waiting time and turnaround time. Existing scheduling algorithms based on priority of the process are not fair and suffer from 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. The Precedence scheduling algorithm with Intelligent Service time is still producing large number of context switches and Factor of Precedence is not balanced among the priority and burst times. This motivated me to design a novel algorithm which uses Balanced Factor of Precedence (BFP) to determine the order of execution. Also it uses Mean average of burst times of processes as its time quantum. This will reduce the number of context switches, average waiting time and average turnaround time. Also attains better response time.

II. Preliminaries

A process is an instance of a computer program that is being executed. The processes waiting to be assigned to a processor are put in a queue called ready queue. The time for which a process holds the CPU is known as burst time. Arrival Time is the time at which a process arrives at the ready queue. The interval from the time of submission of a process to the time of completion is the turnaround time. Waiting time is the amount of time a process has been waiting in the ready queue. The number of times CPU switches from one process to another is known as context switch. The optimal scheduling algorithm will have minimum waiting time, minimum turnaround time and minimum number of context switches.

Basic Scheduling Algorithms

First Come First Serve (FCFS)

In the First-Come-First-Serve (FCFS) algorithm, the CPU is assigned immediately to that process which arrives first at the ready queue. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion.
Shortest Job First (SJF)

In this strategy the scheduler arranges processes with the Burst times in the ready queue, so that the process with low burst time is scheduled first. If two processes having same burst time and arrival time, then FCFS procedure is followed.

Shortest Remaining Time First (SRTF)

This is same as the SJF with pre-emption, which small modification. For scheduling the jobs system need to consider the remaining burst time of the job which is presently executed by the CPU also along with the burst time of the jobs present in the ready queue.

Priority Scheduling Algorithm

It provides the priority to each process and selects the highest priority process from the ready queue.

Round robin Scheduling Algorithm

Round Robin (RR) is one of the oldest, simplest, and fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. Here every process has equal priority and is given a time quantum after which the process is preempted.

III. Related Work

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

IV. Proposed Algorithm

The Newly proposed algorithm overcomes the problem of starvation since it considers both priority as well as burst time equally called balanced factor of precedence to compute the order of execution.

Using,  \[ BFP_i = 0.5 * P_i + 0.5 * BT_i \]

Where  \( BT \) is assigned in such a way that shorter process gets higher number.\( P \) denotes Priority of the Process.

To produce better response time, the algorithm shares the time quantum among the processes. It is best suited for time shared systems. The time quantum is chosen as the mean average of burst times of the processes. Due to this, it does not produce poor response time and also it will not have more number of context switches. That is why the proposed algorithm produces better results.

The algorithm is as follows:

1. Start
2. The Burst times, Priority of each process are taken as input
3. Now Assign value of \( BT \) in such a way that shorter process gets higher number.
4. Now Calculate Balanced Factor of Precedence (BFP) as follows
\[ BFP_i = 0.5 * P_i + 0.5 * BT_i \]

For each process in the Queue. Where \( P \) denotes priority and \( BT \) is the value denoted in step 3

5. Now order of execution (OE) is assigned to processes such that the process with highest BFP will be executed first. If there are any ties in BFP values then it will be resolved by considering process with highest priority.

6. Once the execution sequence is found, Now compute the time quantum using
\[ TQ = \sum \text{Burst time}_i / N \]

Where \( N \) represents Number of Processes
i.e Time quantum is the Mean Average of the Burst times of processes in the Ready Queue.

7. Assign \( TQ \) to process \( P_i \) for each \( i \)
8. Repeat step 7 until the ready queue becomes empty.
9. Stop

V. Experiments and Illustration

A. 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.

**B. Illustration and Results**

**Example**: Suppose there are five processes. Burst time and arrival time have been assumed to be in milliseconds. We calculate ‘BFP’ to find the order of execution and assign value of BT in such a way that shorter process gets higher number. Also Mean average is calculated and considered as time quantum.

Let us consider the following processes with their priorities.

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>P2</td>
<td>53</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>P4</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>P5</td>
<td>20</td>
<td>1</td>
</tr>
</tbody>
</table>

Now higher number of BT is given to shorter process.

Balanced Factor of Precedence (using $\text{BFP}_i = 0.5 * P_i + 0.5 * \text{BT}_i$) is calculated to find the order of execution. The process with highest FP value will be executed first and process with lowest BFP value will be executed last.

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst time</th>
<th>Priority</th>
<th>BT</th>
<th>BFP</th>
<th>Execution Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>11</td>
<td>3</td>
<td>4</td>
<td>3.5</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>53</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P3</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>4.5</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>41</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P5</td>
<td>20</td>
<td>1</td>
<td>3</td>
<td>1.5</td>
<td>5</td>
</tr>
</tbody>
</table>

Now TQ is Computed as Mean average of Burst times

$\text{TQ} = \frac{\sum \text{Burst time}_i}{n}$

$\text{TQ} = \frac{11+53+8+41+20}{5} = 27$

Let us consider the following processes with their execution times:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>P1</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>P2</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>P4</td>
<td>109</td>
<td>93</td>
</tr>
<tr>
<td>P5</td>
<td>123</td>
<td>93</td>
</tr>
</tbody>
</table>

Number of Context Switches = 6

Average Waiting Time = $\frac{0+8+66+82+73}{5} = 45.8$

Average Turnaround Time = $\frac{8+19+109+123+93}{5} = 70.4$

If we Compare the above results with other variants of RR scheduling algorithm are as follows:
From the above graphs we can observe that the proposed algorithm is producing optimal values in terms of context switches, average turnaround time and average response time.

VI. Conclusion & Future Work

From the experimental results, we can say that the proposed algorithm PSMTQ works better than any other variants of RR scheduling algorithm in terms of decreasing number of Context switches, average waiting time and average turnaround time. Also this algorithm overcomes starvation and gives precedence to those processes with higher priority and lower burst time. Future work can be based on this algorithm that includes processes with different arrival times.

VII. References


AUTHOR

P.Surendra Varma received his M.Tech in Computer science Engineering from Acharya Nagarjuna university campus. He is working as an Assistant Professor in NRI institute of technology, Vijayawada. His research interests includes Bioinformatics, compression techniques, operating systems, theory of computation, compiler design, programming languages, data mining and warehousing, software engineering.