Dynamic Time Slice Calculation for Round Robin Process Scheduling Using NOC

G. Siva Nageswara Rao¹, N. Srinivasu², S.V.N. Srinivasu³, G. Rama Koteswara Rao⁴
¹²Department of Computer Science and Engineering, K L University, India
³Department of Computer Science, HOD, PNC&KR PG College Narasarao PET, India
⁴Department of Computer Science and Engineering, V R Siddhatha Engg. College, Vijayawada, India

Article Info

Article history:
Received Apr 12, 2015
Revised Jul 20, 2015
Accepted Aug 16, 2015

Keyword:
Number of context switches
Number of cycles
Remaining burst time
Time slice
Turnaround time

Corresponding Author:
G. Siva Nageswara Rao,
Department of Computer Science and Engineering,
K L University,
Greenfields, Vaddeswaram, Guntur District, Andhra Pradesh 522502.
Email: sivanags@kluniversity.in

1. INTRODUCTION

Scheduling is fundamental operating system function and scheduling is needed for each and every resource. CPU is one of the major resource so its scheduling occupies major role in the design of operating system [1]-[3]. CPU scheduling deals with problem of deciding which of the process in the ready queue is to be allotted to the CPU for processing. For doing this FCFS, SJF, Priority and Round Robin Algorithms are available [4]-[7].

The following characteristics CPU usages (load the cpu as busy as possible),
1) Throughput (number of processing that complete their execution in unit of time)
2) Turnaround time (amount of time to execute particular process)
3) Waiting time (amount of time the process has been waiting in the ready queue)
4) Response time (amount of time it take from when a request was submitted until the first response is produced, not output) are used compare and determine which algorithm is best.

Round Robin Scheduling is designed for time sharing operating systems, this is very similar to FCFS only difference is preemption is added to it to switch from one process to another process. In round robin algorithm initially time quantum or time slice ,it varies with milliseconds and this is used to preempt i.e. CPU switches from one process to another after each and every time quantum, and sometimes switching occur even before the time completion of time quantum [8]-[9], [17].

Here we keep the ready queue as a circular FIFO queue. The CPU scheduler selects the one of the process from the ready queue and CPU is allotted to it end. When the process schedule one of the two things may happens [13]-[16].

1) The burst time of the process greater than time slice CPU switches to another process which is selected by the CPU Scheduler for the next time quantum, the current process is sent back to the ready queue and it will be observed from the remaining time later [18]-[19], [26].

2) If the burst time of the process is less than the times slice then the current process is removed from the ready queue and the CPU switches another process which is selected by the schedule [20]-[22].

There are numerous researches going around the globe on improving the performance of round robin algorithm. The author of [1] proposes a fuzzy approach to find the suitable time slice for the processes. They have presented the results using different simulations. The novelists of [2] propose a median based approach to find the time slice, combining the conventional shortest job first and Round Robin algorithms. The authors of [3] proposes a new technique using maximum and minimum burst time of the set of processes in the ready queue and calculating a modified time slice. The authors of [4] talks about calculating the time slice using median and highest burst time and then executing the processes as per the new calculated time slice. The researchers of [5] provided a mathematical model for calculating the waiting time and turnaround time. The authors of [6] talks about calculating the mean of the burst times of all the processes and then finds the difference between the mean of the burst time and the burst time of a particular process and allocates the CPU to the process which has the maximum difference.

So, a good scheduling algorithm should possess the following characteristics [2]:
1) Minimize the context switches.
2) Maximize the CPU utilization.
3) Maximize the throughput.
4) Minimize the turnaround time.
5) Minimize the waiting time.
6) Minimize response time.

2. PROPOSED APPROACH

Our proposed approach does not aim to change the behavior of the conventional round robin algorithm but to improve it further. In our proposed approach, we will be modifying the time slice of only those processes which require a slightly greater time than the allotted time slice cycle(s) [1]-[3]. The remaining processes will be executed in the basic Round Robin manner. Hence we calculate the remaining burst time and no. cycles for each process [23]-[25]. Based on the remaining burst time, we sort the process, if the remaining burst time is less than or equal to the one time slice then execute the same process otherwise go for next process. If more than one process having the same remaining burst time then use the Shortest Job First Scheduling Algorithm [10]-[12].

TS: Time Slice
BT : Burst Time
RBT : Remaining Burst time
RBT [Pi] = BT [Pi] % TS
NOC : Number of Cycles
NOC [Pi] = ceil (BT [Pi] / TS), where ceil function gives the largest integer greater than or equal to the number.

2.1. Proposed Algorithm

Step1: START
Step2: Make a ready queue of the Processes say Request.
Step3: Calculate the Time Slice .(TS = floor ((Σ(BT [Pi])/N)))
Step4: Calculate the Remaining burst time and Number of Cycles for all processes (RBT = BT[Pi] % TS, NOC = BT[Pi] / TS)
Step5: Sort the all processes based on remaining burst time & NOC.
Step6: Pick the process from the ready queue and allocate the CPU to it for a Time interval of up to 1 time quantum.
Step7: If the remaining CPU burst time of the currently running process is less than or Equal to the one time quantum then allocate CPU again to the currently running process, otherwise to the next process.

Step8: Repeat Step6 & Step 7 until all process are scheduled.
Step9: END.
Example 1: Consider the Five Process and their Time slice is given in the following Table.
(a) Static Time Slice:
Consider the Static Time Slice = 10

<table>
<thead>
<tr>
<th>Process Name</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS1</td>
<td>12</td>
</tr>
<tr>
<td>PS2</td>
<td>15</td>
</tr>
<tr>
<td>PS3</td>
<td>23</td>
</tr>
<tr>
<td>PS4</td>
<td>37</td>
</tr>
<tr>
<td>PS5</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>Avg. TAT.</th>
<th>Avg. TWT.</th>
<th>NCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Round Robin</td>
<td>61.6</td>
<td>60</td>
<td>13</td>
</tr>
<tr>
<td>Aashna Bisht Method</td>
<td>71.2</td>
<td>49.6</td>
<td>10</td>
</tr>
<tr>
<td>My Proposal1</td>
<td>65.2</td>
<td>40.2</td>
<td>7</td>
</tr>
<tr>
<td>My Proposal2</td>
<td>59.2</td>
<td>37.6</td>
<td>7</td>
</tr>
</tbody>
</table>

In the above Table 1 My Proposal2 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.

(b) Dynamic Time Slice:
TQ = Avg. Burst time of all the process (i.e. TS= 22)

| 0        | PS1 12 | 12 PS2 | 27 27 | PS5 48 | 48 PS3 | 71 71 | PS4 108 |

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>Avg. TAT.</th>
<th>Avg. TWT.</th>
<th>NCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Round Robin</td>
<td>59</td>
<td>40.4</td>
<td>6</td>
</tr>
<tr>
<td>Aashna Bisht Method</td>
<td>58</td>
<td>36.4</td>
<td>5</td>
</tr>
<tr>
<td>My Proposal1</td>
<td>53.2</td>
<td>31.8</td>
<td>4</td>
</tr>
</tbody>
</table>

In the above Table 4 My Proposal1 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.

Example 2: Consider the Five Process and their Time slice is given in the following Table.
(a) Static Time Slice:
Consider the Static Time Slice = 4

<table>
<thead>
<tr>
<th>Name of the Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS1</td>
<td>19</td>
</tr>
<tr>
<td>PS2</td>
<td>9</td>
</tr>
<tr>
<td>PS3</td>
<td>23</td>
</tr>
<tr>
<td>PS4</td>
<td>13</td>
</tr>
<tr>
<td>PS5</td>
<td>17</td>
</tr>
</tbody>
</table>

| 0  | PS2 9 | PS4 22 | PS5 39 | PS1 58 | PS3 81 |

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>Avg. TAT.</th>
<th>Avg. TWT.</th>
<th>NCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Round Robin</td>
<td>59</td>
<td>40.4</td>
<td>6</td>
</tr>
<tr>
<td>Aashna Bisht Method</td>
<td>58</td>
<td>36.4</td>
<td>5</td>
</tr>
<tr>
<td>My Proposal1</td>
<td>53.2</td>
<td>31.8</td>
<td>4</td>
</tr>
</tbody>
</table>

In the above Table 6 My Proposal1 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.
Table 7. Comparison table for various scheduling algorithms using static time slice

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>Avg. TAT.</th>
<th>Avg. TWT.</th>
<th>NCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Round Robin</td>
<td>69.6</td>
<td>52.4</td>
<td>22</td>
</tr>
<tr>
<td>Aashna Bisht Method</td>
<td>60.6</td>
<td>44.4</td>
<td>18</td>
</tr>
<tr>
<td>My Proposal1</td>
<td>57.0</td>
<td>40.8</td>
<td>16</td>
</tr>
<tr>
<td>My Proposal2</td>
<td>57.0</td>
<td>40.8</td>
<td>16</td>
</tr>
</tbody>
</table>

(b) Dynamic Time Slice:

TS = Avg. Burst time of the all the process (i.e. TS = 16)

Table 8. Comparison table for various scheduling algorithms using Dynamic time slice

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>Avg. TAT.</th>
<th>Avg. TWT.</th>
<th>NCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Round Robin</td>
<td>62.6</td>
<td>46.4</td>
<td>7</td>
</tr>
<tr>
<td>Aashna Bisht Method</td>
<td>51.2</td>
<td>35.6</td>
<td>5</td>
</tr>
<tr>
<td>My Proposal1</td>
<td>41.8</td>
<td>25.6</td>
<td>4</td>
</tr>
</tbody>
</table>

3. RESULTS & GRAPH

Example 1:

Figure 1. Comparison table for various scheduling algorithms using static time slice

In the above Figure 1 My Proposal2 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.

Figure 2. Comparison table for various scheduling algorithms using Dynamic time slice

In the above Figure 2 My Proposal2 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.
In the above Figure 2, My Proposal1 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.

Example 2:

![Comparison table for various scheduling algorithms using static time slice](image)

In the Figure 3, My Proposal2 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high. The authors of [6] talks about calculating the mean of the burst times of all the processes and then finds the difference between the mean of the burst time and the burst time of a particular process and allocates the CPU to the process which has the maximum difference.

![Comparison table for various scheduling algorithms using Dynamic time slice](image)

With the reference of the [6], we modified the algorithm with better results than earlier scheme scheduling process. We scheme takes lower time than the basic round robin method, which is shown in the Figure 4. In the above Figure 4 My Proposal1 Algorithm has minimum Avg. TAT, minimum TWT and minimum no. of Context switches. Due to less context switching, the processor idle time is low and resource utilization is very high.

4. CONCLUSION

In this paper an improvement for the conventional round robin algorithm is proposed which is being supported by a set of hypothetical examples and a better amount of improvement is observed. The approach can be further refined using the concept of arrival time.
REFERENCES


Dynamic Time Slice Calculation for Round Robin Process Scheduling ... (G. Siva Nageswara Rao)