AN EFFICIENT ROUND ROBIN CPU SCHEDULING ALGORITHM USING DYNAMIC TIME SLICE

G. Siva Nageswara Rao, S.V.N. Srinivasu,

Department of Computer Science and Engineering, K L University,
Department of Computer Science, HOD, PNC & KR PG College Narasaraopet.

Received on: 18.10.2016
Accepted on: 11.11.2016

Abstract

Conceptual CPU Scheduling is one of the basic ideas of Operating System. Round Robin (RR) CPU planning calculation is ideal CPU booking calculation in timeshared frameworks. The execution of the CPU relies on upon the determination of time quantum in timeshared frameworks. The time quantum taken in RR calculation is static that reductions the execution of CPU. In this paper determination of time quantum is talked about and another CPU planning calculation for timeshared frameworks is proposed and is called as EDRR (Efficient Dynamic Round Robin) calculation. The goal of this paper is to roll out an improvement in round robin CPU booking calculation so that the execution of CPU can be moved forward. EDRR additionally incorporates points of interest of round robin CPU booking calculation of less risk of starvation. Round robin CPU planning calculation has high setting switch rates, huge reaction time, expansive holding up time, huge turnaround time and less throughput, these hindrances can be enhanced with new proposed CPU booking calculation. In this paper investigation of number of connection switches, the normal holding up time and the normal turnaround time of procedures in round robin CPU planning calculation, SRBRR (Shortest Remaining Burst Round Robin), ISRBRR (Improved Shortest Remaining Burst Round Robin) and new proposed EDRR CPU booking calculation has been finished.

Keywords: Turnaround time, Waiting time, round robin, CPU scheduling, context switching.

1. Introduction

Central Processing Unit (CPU) is the heart of the PC framework so it ought to be used effectively. For this reason, CPU planning is critical. CPU Scheduling is one of the fundamental ideas of Operating System. Booking of PC assets between numerous procedures is essential [1].
CPU scheduling algorithms are:

- **FCFS (First Come, First Serve) CPU Scheduling**: The process that requests the CPU first is allocated to CPU first.

- **SJF (Shortest Job First) CPU Scheduling**: The process with the shortest CPU burst time is allocated to CPU first.

- **Priority Scheduling**: The process with high priority is allocated to CPU first.

- **Round Robin Scheduling**: Is used in timesharing systems. It is same as FCFS scheduling with pre-emption added to switch between processes. A static Time Slice (TS) is used in this CPU Scheduling.

The different scheduling parameters [2][3] for the selection of the scheduling algorithm are:

- **Context Switch**: Context switch is a procedure of putting away and re-establishing settings (state) of a pre-empted process, with the goal that execution can be continued from the same point at a later time. Setting exchanging is wastage of time and memory that prompts the enhance in the overhead of scheduler, so the objective of CPU booking calculations is to advance just these switches.

- **Throughput**: Throughput is characterized as the number of processes finished in a timeframe. Throughput is less in round robin booking. Throughput and setting exchanging are conversely relative to each other.

- **CPU Utilization**: Is characterized as the part of time CPU is being used. More often than not, the augment the CPU use is the point of the CPU planning.

- **Turnaround Time**: Is characterized as the aggregate time which is spent to finish the procedure and is to what extent it requires the investment to execute that procedure.

- **Waiting Time**: Is characterized as the aggregate time a procedure has been holding up in prepared line.

- **Response Time**: Is preferable measure over turnaround time. Reaction time is characterized as the time utilized by the framework to react to a specific procedure. Consequently, the reaction time ought to be as low as could be expected under the circumstances.

Different characteristics of good scheduling algorithm are [3]:

- Minimum context switches.

- Maximum CPU utilization.
- Maximum throughput.
- Minimum turnaround time.
- Minimum waiting time.

This paper related work is discussed in Section 2. Section 3 explains Round Robin Scheduling Algorithm. Section 4 introduces the Proposed Algorithm with example. Section 5 will provide conclusion of the work.

II. Previous Work

The RR scheduling algorithm has disadvantage that it uses static time slice (TQ). Many research works have been done to improve the performance of the RR scheduling algorithm [18]. A new approach SAAR [4] algorithm uses dynamic time quantum which is repeatedly adjust by the burst time of running processes. SMDRR [5] [12] [13] [16] algorithm uses sub contrary mean or harmonic mean for dynamic time quantum. Similarly SRBRR [6] [14] [15] [17] algorithm uses median for dynamic time quantum and a new approach for time quantum is equal to ceil (sqrt(median * highest_burst_time)) is used in ISRBRR [7][10][11][19][20].

III. Round Robin Scheduling Algorithm

The RR scheduling algorithm [8][9] is given by following steps: -

Step 1:

The scheduler maintains a queue of ready processes and a list of blocked and swapped out processes.

Step 2:

The Process Control Block of newly created process is added to end of ready queue. The Process Control Block of terminating process is removed from the scheduling data structures.

Step 3:

The scheduler always selects the Process Control Block from the head of the ready queue.

Step 4:

When a running process finishes its time quantum, then it is moved to end of ready queue

Step 5:

The event handler performs the following actions:

a) When a process makes an input -output request or swapped out, its Process Control Block is removed from ready queue to blocked/swapped out list.
b) When I/O operation awaited by process is swapped in its Process Control Block or a process finishes is removed from blocked/swapped list to end of ready queue.

There are some drawbacks [9] of round robin CPU scheduling algorithm for operating system which are as follows:

- Static time slice
- Larger waiting time and Response time
- Large number of context switches
- Low throughput

So it can be concluded that the round robin algorithm is not suitable for real time systems. The proposed new algorithm can be used for real time system which is described in next section.

IV. Proposed Algorithm

The proposed CPU Scheduling calculation depends on the little change in round-robin planning calculation. It executes the briefest employment first rather than FCFS amid round robin calculation. The proposed dispenses with on the hindrance of round robin calculation in which procedures are planned in first started things out serve way. This round robin calculation is not reasonable for procedures with short CPU burst. So it enhances the holding up time and reaction time of procedures which diminishes in the framework throughput. The new proposed calculation utilizes the dynamic time cut rather than static time cut. The proposed engineering evacuates the deformities of executing basic round robin design. The proposed calculation will be executed in two stages which will minimizes various execution parameters, for example, connection switches, the normal turnaround time and the normal holding up time.

This algorithm has following steps:

Step 1:
Processes are arranged in increasing order of their CPU burst time.

Step 2:
Set the time slice is equal to the CPU burst time of the first process (The shortest process).

Step 3:
Calculate the median and mean of CPU burst time of all the processes

Step 4:
Set the time slice(TQ) according to following method-If
(mean>median)

\[ TQ = \text{ceil} \left( \sqrt{(\text{mean} \times \text{highest burst time}) + (\text{median} \times \text{lowest burst time})} \right) \]

Else If (median>mean)

\[ TQ = \text{ceil} \left( \sqrt{(\text{median} \times \text{highest burst time}) + (\text{mean} \times \text{lowest burst time})} \right) \]

Else

\[ TQ = \text{mean} \]

Step 5:

Allocate CPU to every process according to the round robin.

If remaining burst time is less than one-time quantum execute the same process otherwise go to step 6

Step 6: Go to step 1.

Example: CASE 1 :(Process Without Arrival Time)

Consider the Ten Processes named P1 to P10 and with their CPU burst time

**Table 1: Input Table**

<table>
<thead>
<tr>
<th>Process Name</th>
<th>CPU Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>23</td>
</tr>
<tr>
<td>P2</td>
<td>75</td>
</tr>
<tr>
<td>P3</td>
<td>93</td>
</tr>
<tr>
<td>P4</td>
<td>48</td>
</tr>
<tr>
<td>P5</td>
<td>2</td>
</tr>
<tr>
<td>P6</td>
<td>5</td>
</tr>
<tr>
<td>P7</td>
<td>12</td>
</tr>
<tr>
<td>P8</td>
<td>20</td>
</tr>
<tr>
<td>P9</td>
<td>26</td>
</tr>
<tr>
<td>P10</td>
<td>34</td>
</tr>
</tbody>
</table>

Round Robin Scheduling: Time Slice is 30

**Fig. 1: Gantt Chart for RR Algorithm.**

**Fig. 2: Gantt Chart for Proposed Algorithm.**
Table 2: Comparison of RR, IDRR, MDRR and Proposed Algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Waiting Time</th>
<th>Avg. Turnaround Time</th>
<th>No. of Context Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>194</td>
<td>160</td>
<td>15</td>
</tr>
<tr>
<td>IDRR</td>
<td>162</td>
<td>151</td>
<td>11</td>
</tr>
<tr>
<td>MDRR</td>
<td>118</td>
<td>84</td>
<td>11</td>
</tr>
<tr>
<td>Proposed(EDRR)</td>
<td>109</td>
<td>75</td>
<td>9</td>
</tr>
</tbody>
</table>

CASE 2: (Process with Arrival Time)

Table 3: Input Table.

<table>
<thead>
<tr>
<th>Process Name</th>
<th>CPU Burst Time</th>
<th>Arrival Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>93</td>
<td>12</td>
</tr>
<tr>
<td>P4</td>
<td>48</td>
<td>20</td>
</tr>
<tr>
<td>P5</td>
<td>2</td>
<td>23</td>
</tr>
<tr>
<td>P6</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>P7</td>
<td>12</td>
<td>34</td>
</tr>
<tr>
<td>P8</td>
<td>20</td>
<td>48</td>
</tr>
<tr>
<td>P9</td>
<td>26</td>
<td>75</td>
</tr>
<tr>
<td>P10</td>
<td>34</td>
<td>93</td>
</tr>
</tbody>
</table>

Fig. 3: Gantt Chart for RR Algorithm.

Fig. 4 Gantt Chart for Proposed Algorithm.

Table 4: Comparison of RR, IDRR, MDRR and Proposed Algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Waiting Time</th>
<th>Avg. Turnaround Time</th>
<th>No. of Context Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>173</td>
<td>140</td>
<td>15</td>
</tr>
<tr>
<td>IDRR</td>
<td>147</td>
<td>130</td>
<td>11</td>
</tr>
<tr>
<td>MDRR</td>
<td>97</td>
<td>69</td>
<td>11</td>
</tr>
<tr>
<td>Proposed(EDRR)</td>
<td>93</td>
<td>55</td>
<td>9</td>
</tr>
</tbody>
</table>

Results:
V. Conclusion

This paper demonstrates another CPU planning calculation. Correlation of various calculations i.e. round robin, SRBRR, ISRBRR, IDRR, MDRR and the proposed calculation EDRR has been finished. It is inferred that the proposed calculation is more effective than round robin calculation as a result of it has less normal holding up time, less normal turnaround time and less number of connection changes when contrasted with round robin, so it lessens the working framework overhead. The proposed calculation is the mix of the most limited employment first CPU booking calculation and the round robin CPU planning calculation with productive and element time cut.

References

3. Ajit Singh, Priyanka Goyal, SahilBatra,” An Optimized Round Robin Scheduling Algorithm for CPU


