Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum

Pallab Banerjee, Probal Banerjee, Shweta Sonali Dhal

Abstract:- Round Robin Scheduling algorithm is designed especially for time sharing Operating system (OS). It is a preemptive CPU scheduling algorithm which switches between the processes when static time Quantum expires. The Round Robin Scheduling algorithm has its disadvantages that is its longer average waiting time, higher context switches, higher turnaround time. In this paper a new algorithm is presented called Average Max Round Robin (AMRR) scheduling algorithm. In this scheduling algorithm the main idea is to adjust the time Quantum dynamically so that (AMRR) perform better performance than simple Round Robin scheduling algorithm.

Keywords-Operating System, Round Robin, Average Max Round Robin, Turnaround time, Waiting time, Context Switch.

I. INTRODUCTION

An operating system is a system software which makes an interface between user and computer hardware. Operating system provides a platform in which user can interact with hardware and execute programs in an efficient manner. Modern operating system and time sharing system are more complex, they have evolved from a single task to multitasking environment in which processes run in synchronized manner. In a multiprocessing and multitasking environment if several processes are ready to run at the same time, the system must choose among them and assigned to run on the available CPUs, is called CPU scheduling. Allocating CPU to a process requires careful awareness to assure justice and avoid process starvation for CPU. Scheduling decision try to reduce the following: turnaround time, waiting time, context switches. CPU scheduling algorithm decides which of the processes in the Ready Queue sets a timer to interrupt after a fixed time quantum (TQ). The CPU scheduler goes around the Ready Queue and allocates the CPU to each processes by the help of Dispatcher for a time interval of up to the Time Quantum(TQ). If new process arrives then it is added to the tail of Circular Queue. The CPU scheduler picks the first process from the Ready Queue sets a timer to interrupt after a fixed time quantum (TQ) and dispatches the process. After TQ is expired, the process is preempted and added to the tail of the Circular Queue. The process finishes before the end of the TQ, the process itself preempt the CPU willingly. In this paper, we tried to solve the Time Quantum problem by adjusting the Time Quantum Dynamically with respect to the existed set of processes in Ready Queue.

II. PRELIMINARIES

Program is refer to the set of instruction that are executed in pipeline fashion. Program in execution is called process. Process are represented by Process Control Block (PCB). PCB contains many information about process such as process state, process number, program counter, list of open files, registers and CPU scheduling information. When process enter into the main memory and are ready and waiting to execute are kept in the data structure called Ready Queue. When a process assign to the CPU, it execute or while waiting for some event to occur. The process which are waiting for I/O request are kept in Device Queue. The Long term scheduler or job scheduler select process from job pool and load them into main memory for execution. Short term scheduler or CPU scheduler select from among the processes that are ready to execute and allocates the CPU to one of them. Medium term scheduler is used in time sharing system. The main advantage of Medium term scheduler is sometimes
Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum

it remove processes from main memory and thus reduce degree of multiprogramming. Later the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called as swapping. So, the process is swapped out, and is later swapped in, by the medium term scheduler [1].

The scheduler is mainly concerned with:

Arrival time: - The time at which process arrives in main memory.

Burst time: - The time for which a process holds the CPU.

Waiting time: - The amount of time, process waiting in the Ready Queue.

Turnaround time: - The total time between arrival of a process and process completion.

Response time: - The time from the submission of a request by a process till its first response.

III. RELATED WORK

In the last few years different approaches are used to increase the performance of Round Robin scheduling like Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice [4], Multi-Dynamic time Quantum Round Robin (MDTQRR) [5]. Min-Max Round Robin (MMRR) [3], Self-Adjustment Time Quantum in Round Robin (SARR) [8], Dynamic Quantum with Re-adjusted Round Robin (DQRRR)[11].

IV. PROPOSED APPROACH

Let’s assume that the burst time of the processes is taken as sorted increasing order so that it will give better turnaround time and waiting time. Generally in Round Robin algorithm the performance depends upon the size of fixed or static Time Quantum (TQ). If TQ is too large then Round Robin algorithm approximate to First Come First Served (FCFS ). If the Time Quantum is too small then there will be many context switching between the processes. So, our approach solved this problem by taking a dynamic TQ. Where TQ is the mean of the summation of the average and the Maximum Burst time.

\[
\text{AVG} = \frac{\text{Summation of Burst Time of all the processes}}{\text{Number of processes}}
\]

\[
\text{TQ} = \frac{\text{AVG} + \text{MAXBT}}{2}
\]

V. PROPOSED ALGORITHM

In our proposed algorithm, processes are already present in the Ready Queue (RQ). By default, Arrival Time (AT) is assigned to zero. The number of processes ‘n’ and CPU Burst Time (BT) are accepted as input and Average Turnaround Time (ATT), Average Waiting Time (AWT) and number of Context Switch (CS) are produced as output. Let TQ and TQn be the time quantum and new time quantum respectively. The pseudo code for the algorithm is presented in Figure 1 and the flowchart of the algorithm is presented in Figure 2.

VI. PERFORMANCE METRICS

The proposed algorithm is designed to meet all scheduling criteria such as maximum CPU utilization, maximum throughput, minimum turnaround time, minimum waiting time and context switches. Here we are considering three performance criteria in each case of our experiment.

1. Turnaround Time(TAT)

\[
\text{TAT} = \text{Finish Time} - \text{Arrival Time}
\]

Average Turnaround Time should be less.

2. Waiting Time(WT)

\[
\text{WT} = \text{Start Time} - \text{Arrival Time}
\]

Average Waiting Time should be less.

3. Context Switch

The number of context Switch should be less

VII. ILLUSTRATION

Suppose four processes arriving time =0,and CPU burst time is (P1=70, P2=65,P3=10, P4=15).Then the processes are sorted in ascending order which results in sequence P3=10,P4=15,P2=65 ,P1=70. Then TQ is calculated. Where TQ is the mean of the summation of the average and the Maximum Burst time i.e. AVG=(10+15+65+70)/4=40. So TQ is equal to (AVG+MAXBT)/2 i.e. TQ=(40+70)/2=55. After first iteration the remaining CPU burst time sequence is P3=0,P4=0,P2=10,P4=15.In this case, processes P3 and P4 are deleted from the Ready Queue. Again CPU burst time is sorted in ascending order and new TQ is calculated. Here new TQ is equal to 14. After second iteration the remaining CPU burst time sequence is P2=0 and P4=1. Then P2 is deleted from RQ. After third iteration the remaining CPU burst time sequence is P4=0. Since, now there is no process in the RQ, it completes its execution and ATT,AWT and CS are calculated. In this case, ATT=98.75,AWT=58.75,CS=5.

VIII. EXPERIMENTAL ANALYSIS

In every case we will compare the result of the proposed AMRR method with Round Robin scheduling algorithm. Here we have taken 20 as the static time quantum (TQ) for RR algorithm

CASE 1:-

Let’s consider four processes (P1,P2,P3,P4) with arrival time =0 and burst time(8,40,72,84) as shown in the Table 1. Table 2 shows the output using RR algorithm and AMRR algorithm. Figure 3 and Figure 4 shows Gantt chart of both RR and AMRR algorithm respectively.
1. All the processes in the ready queue are stored in ascending order
   // n = Number of processes
   // i = Loop variable (counter)
   // BT = Burst Time

2. while (RQ != NULL)
   // RQ = Ready Queue
   // TQ = Time Quantum
   // AVG = Average of all the processes
   // MAXBT = Maximum Burst Time
   AVG = (Sum of BT of all the processes) / Number of processes.
   // (Use round off function in AVG.)
   TQ = (AVG + MAXBT) / 2
   // (Use round off function in TQ)
   // (Remaining Burst time of the process)
   // If one process is there then TQ is equal to BT itself

3. // Assign TQ to (1 to n) processes
   for i = 0 to n loop
   { 
     Pi -> TQn
   }
   // TQn = New Time Quantum
   end of for
   // Assign TQn to all the available processes

4. Calculate the remaining Burst time of the processes.

5. If (new process arrived and BT != 0 or new process is arrived and BT == 0)
   then go to step 1
   else if (new process is not arrived and BT != 0)
   then go to step 2
   else
   go to step 6
   end of if
   end of while

6. Calculate ATT, AWT, CS
   // ATT = Average Turnaround time
   // AWT = Average waiting time
   // CS = Number of context switch

7. End

Figure 1. Pseudo code for Avg Max Round Robin (AMRR) algorithm

Figure 2. Flowchart of Avg Max Round Robin
Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum

Table 1. Processes with Burst Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 2. Comparison between RR algorithm and our new proposed AMRR algorithm (CASE 1).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Quantum</th>
<th>Turnaround Time</th>
<th>Waiting Time</th>
<th>Context Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>20</td>
<td>120</td>
<td>69</td>
<td>10</td>
</tr>
<tr>
<td>AMRR</td>
<td>68,13,3</td>
<td>112</td>
<td>61</td>
<td>5</td>
</tr>
</tbody>
</table>

CASE 2:-
Let’s consider four processes (P1, P2, P3, P4) with arrival time = 0 and burst time (41, 42, 43, 44) as shown in the Table 3. Table 4 shows the output using RR algorithm and AMRR algorithm. Figure 5 and Figure 6 shows Gantt chart of both RR and AMRR algorithm respectively.

Table 3. Processes with Burst Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 4. Comparison between RR algorithm and our new proposed AMRR algorithm (CASE 2).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Quantum</th>
<th>Turnaround Time</th>
<th>Waiting Time</th>
<th>Context Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>20</td>
<td>165</td>
<td>122.5</td>
<td>11</td>
</tr>
<tr>
<td>AMRR</td>
<td>43,1</td>
<td>165</td>
<td>62.5</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 3: Gantt chart of RR from Table 1 (CASE 1).

Fig. 4: Gantt chart of AMRR from Table 1 (CASE 1).

Fig. 5: Gantt chart of RR from Table 3 (CASE 2).

Fig. 6: Gantt chart of AMRR from Table 3 (CASE 2).
CASE 3:
Let’s consider four processes (P1, P2, P3, P4) with arrival time = 0 and burst time (25, 50, 180, 200) as shown in the Table 5.

Table 5: Processes with Burst Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>180</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 6 shows the output using RR algorithm and AMRR algorithm. Figure 7 and Figure 8 shows Gantt chart of both RR and AMRR algorithm respectively.

Table 6: Comparison between RR algorithm and our new proposed AMRR algorithm (CASE 2).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Quantum</th>
<th>Turnaround Time</th>
<th>Waiting Time</th>
<th>Context Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>20</td>
<td>277.5</td>
<td>173.75</td>
<td>22</td>
</tr>
<tr>
<td>AMRR</td>
<td>157, 38, 5</td>
<td>241.75</td>
<td>128</td>
<td>5</td>
</tr>
</tbody>
</table>

CASE 4:
Let’s consider four processes (P1, P2, P3, P4) with arrival time (0, 4, 20, 25) and burst time (10, 30, 70, 85) as shown in the Table 7. Table 8 shows the output using RR algorithm and AMRR algorithm. Figure 9 and Figure 10 shows Gantt chart of both RR and AMRR algorithm respectively.

Table 7: Processes with Burst Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>P2</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>P3</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>P4</td>
<td>25</td>
<td>85</td>
</tr>
</tbody>
</table>

Table 8: Comparison between RR algorithm and our new proposed AMRR algorithm (case 4).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Quantum</th>
<th>Turnaround Time</th>
<th>Waiting Time</th>
<th>Context Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>20</td>
<td>101.5</td>
<td>52.75</td>
<td>10</td>
</tr>
<tr>
<td>AMRR</td>
<td>67, 15, 3</td>
<td>93.25</td>
<td>44.5</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig.7: Gantt chart of RR from Table 5 (CASE 3).

Fig.8: Gantt chart of AMRR from Table 5 (CASE 3).
Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum

CASE 5:-
Let’s consider four processes(P1,P2,P3,P4) with arrival time (0,18,30,50) and burst time (25,55,80,100) as shown in the Table 9.

Table 9. Processes with Burst Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>P2</td>
<td>18</td>
<td>55</td>
</tr>
<tr>
<td>P3</td>
<td>30</td>
<td>80</td>
</tr>
<tr>
<td>P4</td>
<td>50</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 10. Comparison between RR algorithm and our new proposed AMRR algorithm (case 5).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Quantum</th>
<th>Turnaround Time</th>
<th>Waiting Time</th>
<th>Context Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>20</td>
<td>156.75</td>
<td>89.25</td>
<td>11</td>
</tr>
<tr>
<td>AMRR</td>
<td>83.17</td>
<td>106.75</td>
<td>41.75</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig.10: Gantt chart for AMRR in Table 7(case 4).

Fig.11: Gantt chart for RR in Table 9(case 5)

Fig.12: Gantt chart for AMRR in Table 9(case 5).

Fig.13: Comparison of average turnaround time of RR and AMRR taking arrival time into consideration.

Fig.14. Comparison of average waiting time of RR and AMRR taking arrival time into consideration.
Fig. 15. Comparison of Context Switches of RR and AMRR taking arrival time into consideration.

REFERENCES