International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 9 Issue 5 ǁ 2021 ǁ PP. 72-76

Modified Round Robin Algorithm based on Priority and Shortness components

Ritik Kumar
VIT University, Vellore

Muskan Agarwal
VIT University, Vellore

Dishit Duggar
VIT University, Vellore

Abstract
One of the critical aspects of designing any Operating System is the type of Scheduling algorithm it supports. These algorithms are used by the by the operating system to allot CPU processing time to each of the process in the ready queue. Round robin algorithm is one such algorithm which is very commonly used and uses the concept of time quantum to allow processes to utilize CPU for a specific amount of time, before moving on to the next process.
In this paper, we intend to analyse and compare a modified version of Round Robin algorithm with the conventional algorithms like FCFS, SJF and RR based on various factors. The modified version of RR uses dynamic time quantum calculated based on a priority component and a shortness component of the process.

Keywords: *Scheduling Algorithms, Round Robin (RR), Time Quantum (TQ), Priority, Ready queue, First Come First Serve (FCFS), Shortest Job First (SJF), Average Waiting Time (ATT), Average Turnaround Time (ATT), Context Switches, Intelligent Time Slice.*

Date of Submission: 16-05-2021 Date of acceptance: 31-05-2021

I. INTRODUCTION

Choosing the right algorithm for a multiprocessor system is a key aspect of designing an efficient operating system. There are various algorithms that work on different aspects. Many algorithms are designed based on the priority of the processes. [1] Priorities can be either dynamic or static. Static priorities are allocated during creation, whereas dynamic priorities are assigned depending on the behaviour of the processes while in the system. To illustrate, the scheduler could favour input/output (I/O) intensive tasks, which lets expensive requests to be issued as soon as possible.

[2] In case of priority based Round Robin scheduling algorithm when similar priority jobs arrive, the processes are executed based on FCFS. The processes have a specific burst time associated. A time quantum is set, and the processes utilize the resources available for that time quantum only. Once the time quantum is crosses, the control is passed to the next process.

Here, we propose a new CPU scheduling algorithm based on dynamic time quantum. This algorithm builds upon the shortcomings of the existing algorithms, taking into account, and reducing the waiting time, turnaround time, number of context switches and the throughput. The proposed algorithm is pre-emptive in its execution and uses the priorities of the associated processes. The result of the proposed algorithm is then compared to that of those obtained by some existing algorithms.

1.1 EXISTING APPROCH

- First Come First Server (FCFS) [3] is one of the simplest scheduling algorithms that works on the simple concept of arrival times of the process. The process that arrives first will be served first and it is non-pre-emptive method of scheduling. A FIFIO queue is used to maintain the arriving processes. In case two or more processes arrive at the same time, so the process that enters the FIFO queue first is executed first. Any process that is in the ready state is put in the queue according to their arrival time.
- SJF [3] algorithm is a form of priority scheduling. Here, the priority associated with each process is the burst time of the process. Therefore, the process with the shortest burst time gets the highest priority. The CPU

is then allocated to the process with the highest priority first, followed by other processes in the same order. In case, if two or more process have the same priority (burst time), then the processes follow the FCFS algorithm.

- In Round Robin algorithm [3], the concept of time-sharing is used. The underlying method is same as the FCFS, that is, the process arriving first is allotted the CPU first, but the difference comes where the algorithm is pre-emptive, that is it does not wait for the completion of the process and moves on to the next process after a certain allotted time (time-quantum). This time-quantum is typically in the range of 1-100 milliseconds. After this allotted time runs out, the current process is delayed, irrespective of whether it is completed or not, and the CPU moves on to the next process in the ready queue. If a process has a burst time less than the time quantum, then the CPU is released after the process has finished so that CPU can be allotted to the next processes in the queue. Conversely, if a process has burst time greater than the time-quantum, the process is delayed after the time slice is expired and then queues back to the position of the tail of the ready queue and wait for it turn again.

II. LITERATURE SURVEY

Extensive work has already been done in the field of scheduling algorithms. Many attempts have been made in order to improve and optimize the existing approaches, as well as come up with new algorithms.

Muhammad Akhtar [4] proposed a hybrid algorithm, comprising of two algorithms – the Shortest Job First and the constrained on the remaining burst time of running process. The underlying concept of the algorithm is that on start it selects the job having the shortest burst time and starts the execution. Then, when a new job arrives in the ready queue, it decides whether to pre-empt the running process or not. If the remaining execution time of the running process is less than or equal to the burst time of the newly arrived process, then the running process will not pre-empt. Otherwise, the running process will be delayed, and the newly arrived process will be allotted the CPU for execution.

Tri Dharma Putra [5] discussed an efficient pre-emptive shortest job first algorithm with lower average waiting time and turnaround time. Three case studies are discussed to understand this algorithm. Here, the ready queue is organised according to the burst time of process. Those routines which require small amount of time for its completion are put in the front of the queue. When a process comes in the ready queue, then the process which is running is pre-empted and the CPU is next allotted to the process having the smallest burst time. After its completion, it is terminated and removed from the waiting process list.

Md Gulzar [6] proposed a self-regulated priority based round robin scheduling algorithm that uses different method to calculate time quantum. First, a time quantum is calculated as the average of all CPUs burst time of processes in the ready queue. Then, the processes are sorted in ascending order and assign high priority for short process. The CPU is then allocated to the processes having the higher priority first. Also, since it follows the pre-emptive method, if the burst time is bigger than the time quantum, then the process is pre-empted, and the CPU is given to the next higher priority process.

Abdulaziz A Alsulami [7] did a performance evaluation of different types of Dynamic Round Robin Algorithm – Adaptive RR, Best Time Quantum RR, Optimal RR using Manhattan Distance Algorithm, and Improved RR, taking into account four factors – Average Waiting Time (AWT), Average Turnaround Time (ATT), Average Response Time (ART) and Number of Context Switches (NCS). Simulation results shows that both Adaptive RR and Optimal RR are more efficient CPU scheduling techniques than the other two algorithms.

III. LIMITATIONS OF ROUND ROBIN ALGORITHM

Round Robin scheduling algorithm has many disadvantages, which are as following:

- *High Average Waiting Time*

For round robin architecture, the process spends the time in the ready queue waiting its turn to own the processor. Due to the presence of time quantum, processes are pushed to leave the processor and return to the waiting state. This procedure produces a high average waiting time, which presents the main disadvantage.

- *Low throughput*

Throughput present the number of process completed per time unit. Due to its time slice, Round Robin is characterized by a high number of context switches, which leads to overall degradation of the system performance (throughput).

- *Context switch*

Ones the time slice finished; the process is forced to leave the CPU. The scheduler stores the context of the current process in stack or a register and allots the CPU to the next process in the ready queue. This concept is known by context switching which leads to time wastage and scheduler overhead.

- *High response time*

Response time is defined as the time between submission of a request and the first CPU response. In general, round robin made larger response time, which causes system performance degradation. To achieve high performance, the reduction of response time shall be indispensable.

- *Very high turnaround time*

Turnaround time is the time between submission of a process and its completion. Round Robin scheduling algorithm is characterized by a high turnaround time. So as a result, to improve the system performance this parameter should be reduced.

IV. PROPOSED APPROACH

The proposed algorithm builds upon the shortcomings of the various existing scheduling algorithms. The factors that it tries to improve are the waiting time, turnaround time, number of context switches and the throughput. It works on the concept of dynamic time-quantum as opposed to the static time-quantum implemented in the case of conventional RR algorithm. In case of static time-quantum, each process is allotted a specific time slice in which the CPU will execute the given process. The dynamic time-quantum is calculated taking into account the priority as well as the shortness and the burst time of the process at hand. As it is dynamic, the time-quantum changes for each round of CPU allotment for all the processes in the ready queue.

To decide the priority of the processes, a priority component is calculated for each process based on initial priority. A shortness component is also calculated based on the burst times of preceding process.

To complete the computation, a time quantum is initially set. Using priority component, shortness component and burst times, an Intelligent Time Slice is computed for each process. For every round, a new time quantum is then calculated using Shortness component. Thus, the algorithm incorporates the elements of priority scheduling algorithm and Shortest Job First algorithm within Round Robin algorithm. This will reduce the mean WT and mean TAT of the process.

V. ALGORITHM

```
1. The processes are sorted according to priority, a new priority is assigned to each process considering the
   original priority and the shortness of the process.
2. The priority component is calculated in the following manner:
a. For n processes (i ϵ [1, n]),
  i.   Priority[i] = 0 (if new priority is > 2 * n / 3)
  ii.  Priority[i] = 1 (if new priority is > n / 3)
  iii. Priority[i] = 2 (if new priority is >= 1)
o Here, the priority decreases with decrease in numerical value (2 is most important while 0 is least
  important)
3. The shortness component is calculated in the following manner:
a. For n processes (i ϵ [1, n]),
  i.   Shortness[i] = 0 (if burst_time[i] > burst_time[i – 1])
  ii.  Shortness[i] = 1 (if burst_time[i] <= burst_time[i – 1])
o Here, shortness increases with decrease in numerical value (0 is longer while 1 is shorter)
4. The intelligent time slice is calculated for each process as follows:
a. intelligent_time_slice = initial_time_slice[i] + burst_time[i] + priority[i] + shortness[i]
5. The above steps are repeated until all processes are completed.
6. Now
a. If the round_number[j] = 1
  i.   time_quantum[j][i] = intelligent_time_slice[i] (if shortness[i] = 1)
  ii.  time_quantum[j][i] = intelligent_time_slice[i] / 2 (if shortness[i] = 0)
b. If the round_number[j] is not 1
  i.   time_quantum[j][i] = time_quantum[j – 1][i] * 2 (if shortness[i] = 1)
  ii.  time_quantum[j][i] = time_quantum[j – 1][i] * 1.5 (if shortness[i] = 0)
7. Average waiting and average turn-around time are then calculated.
```

VI. RESULT ANALYSIS

The proposed algorithm was implemented using C++ language in an IDE (Codeblocks) to check the performance based on the Average Waiting Time and Average Turnaround Time. Other standard algorithms were also implemented to compare the performance between the different algorithms.

A typical scenario is imitated with four processes for two cases – one with all processes having 0 arrival time, and another having explicit arrival time for each process. In case pf RR algorithm and the proposed algorithm, the time slice selected is 5ms.

Table I: Sample Data Taken

Name	Burst Time (ms)	Priority
A	40	1
B	35	2
C	28	3
D	50	4

- **With zero Arrival time**

Table II: Sample data with 0 arrival time

Name	Burst Time (ms)	Arrival Time(ms)	Priority
A	40	0	1
B	35	0	2
C	28	0	3
D	50	0	4

Table III: Result for sample data with 0 arrival time

Algorithm	A.W.T (ms)	A.T.A.T (ms)
SRTF	49.00	48.25
Priority (Pre-emptive)	54.50	53.75
Priority (Non-Pre-emptive)	54.50	64.5
FCFS	54.50	63.25
RR	94.75	95.5
SJF	52.75	58.5
Modified RR (Proposed)	68.5	57.25

- **With Explicit Arrival time**

Table IV: Sample data with explicit arrival time

Name	Burst Time (ms)	Arrival Time (ms)	Priority
A	40	2	1
B	35	1	2
C	28	4	3
D	50	0	4

Table V: Sample data result with explicit arrival time

Algorithm	A.W.T (ms)	A.T.A.T (ms)
SRTF	48.25	86.50
Priority (Pre-emptive)	53.75	92.00
Priority (Non-Pre-emptive)	64.50	101.50
FCFS	63.25	101.56
RR	95.50	133.75
SJF	58.50	96.75
Modified RR (Proposed)	57.75	95.50

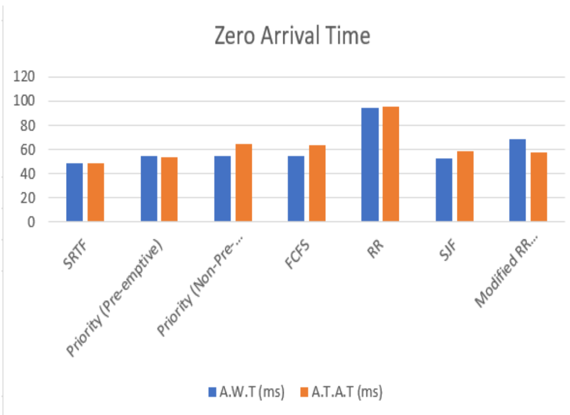

Figure 1: Comparison Graph for zero arrival time

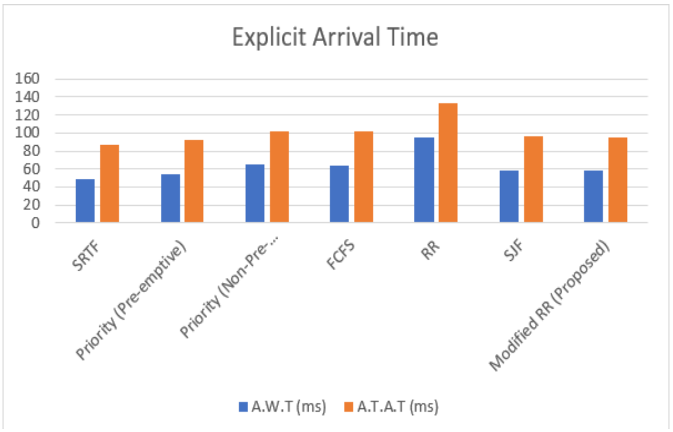

Figure 2: Comparison graph for explicit arrival time

VII. CONCLUSION

From the bar graph we can observe that our approach considerably outperforms the simple RR algorithm. It has lesser average waiting time and turnaround time than Round Robin. While Shortest Job first and shortest remaining time first show better results, they aren't feasible in real time systems. It reduces the overhead and solves the problem of starvation and large time quantum with lesser number of context switches. Thus, we can see that our proposed algorithm meets the demands of a real time system.

REFERENCES

[1]. Chandra Shekar N, Karthik V (2017) "Analysis of Priority Scheduling Algorithm on the Basis of FCFS & SJF for Similar Priority Jobs", International Journal of Engineering Research in Computer Science and Engineering, Vol 4, Issue 3.

[2]. Maniyar bhumi J, Kanani Bhavisha R (2015), "Review on Round Robin Algorithm for Task Scheduling in Cloud Computing", Journal of Emerging Technologies and Innovative Research, Vol 2, Issue 3.

[3]. Andyash Putera Utama Siahaan (2016), "Comparison Analysis of CPU Scheduling: FCFS, SJF and Round Robin", IJDER, Volume 4, Issue 3.

[4]. Muhammad Akhtar, Bushra Hamid, Inayat ur-Rehman, Mamoona Humayun, Maryam Hamayun and Hira Khurshid (2015) "An Optimized Shortest Job First Scheduling Algorithm for CPU Scheduling", Journal of Applied Environmental and Biological Sciences.

[5]. Tri Dharma Putra (2020) "Analysis of Preemptive Shortest Job First (SJF) Algorithm in CPU Scheduling", International Journal of Advanced Research in Computer and Communication Engineering, Vol 9, Issue 4.

[6]. Md Gulzar, K. Ravindra Babu, A Vinaya Babu, S Udaya Kumar (2013) "Self-Regulated Priority based Round Robin Scheduling Algorithm", International Journal of Computer Applications, Page 24-29.

[7]. Abdulaziz A Alsulami, Qasem Abu Al-Haija, Mohammed I Thanoon, Qian Mao (2019), "Performance Evaluation of Dynamic Round Robin Algorithm for CPU Scheduling"