Improved Round Robin Algorithm Under Cloud Computing

Manoj Malik

Maharaja Surajmal Institute Of Technology , Janakpuri , Delhi , India

Hitesh Batra

Maharaja Surajmal Institute Of Technology , Janakpuri , Delhi , India

Sahil Wasan

Maharaja Surajmal Institute Of Technology , Janakpuri , Delhi , India

Abstract

Multiprogramming is a critical component of any operating system. It's a memory-based approach for running many procedures at the same time. In timeshared frameworks, the RR CPU technique is ideal. In systems that rely on timeshared environments, the CPU implementation is dependent on the mode we choose for time quantum. We implemented a progress in the RR method so that CPU execution can be shifted forward, compared it to the original RR scheduling, and demonstrated why our model is more efficient than the original. The purpose of this study is to modify the original RR in such a way that it outperforms the standard RR in terms of waiting and turnaround time. Load balancing is the technique used to utilize cloud computing services efficiently[1].

Keywords : Round robin scheduling, time quantum, burst time, waiting time, turnaround time

I. INTRODUCTION

Cloud computing is a form of parallel and distributed architecture that consists of a collection of interconnected and virtualized PCs. Customers can access cloud information and applications from anywhere at any time, giving customers the advantage of unlimited computing power at a bare minimum cost [2]. The primary benefit of scheduling is that it allows for better performance and system throughput.

The most prevalent pre-emptive scheduling method, referred to as Standard RR (SRR) hereafter is RR scheduling, which is used in real-time operating systems and timesharing [3]. The operating system is driven by a regular interrupt in RR scheduling. For execution, processes are chosen in a predetermined order. The system timer interrupts a process (or CPU burst). Following that interruption, the scheduler switches context to the next process in the ready queue, which is handled as a circular queue. As a result, all requests in the queue are given a chance to be served for a limited time. This scheduling method is commonly used in timesharing systems[4]. The efficiency of the RR algorithm is dependent on the time slice; if the time slice is small, there will be more context changes; if the time slice is long, RR operates similarly to FCFS, with the

potential of process starvation. So, we need to be a better judge of the time slice that we intend upon choosing for our system. Waiting time (i.e., the total time the process spent waiting in the

ready queue), turnaround time (i.e., the total time between process submission and completion), and the number of context shifts all affect the scheduling algorithm's performance. The rate at which the CPU is occupied is referred to as CPU exploitation.

One of the major load balancing metric is throughput[5]. It is the number of procedures completed in a given amount of time. The number of times the process changes context is measured in context switches. This algorithm can be improved by reducing reaction time, WT, and TAT, as well as increasing CPU usage and throughput.

The major focus of this paper relies upon the fact that the fixed quantum time acts as a blocker for improving the algorithms' performance. So, we devised a strategy that focuses on the grassroots concept of burst time of incoming requests instead of defining the riveted quantum time on the basis of a pre-determined value. The paper compares the is assembled as follows. Segment 2 describes the related work in the field. Segment 3 focuses on the proposed solution. Segment 4 provides a piece of conclusive evidence of how our algorithm outperforms the traditional Round Robin algorithm in time-related aspects. Segment 5, finally concludes the paper and describes the future prospects of the work.

II. RELATED WORK

The RR algorithm has the flaw of using static TQ. Several studies have been conducted in recent years to improve the job scheduling process. The below section explores some of the critical works done by other authors.

Adaptive Round Robin approach [4] based on Shortest Burst Time using Smart Time Slice. When the number of processes is odd, Smart Time Slice is equal to the average CPU burst time of all running processes, and when the number of processes is even, time quantum is chosen based on the average CPU burst time of all running processes.

This algorithm [6] proposes a method under the name An Additional Improvement in Round Robin, that focuses on improving the round-robin CPU algorithm. In comparison to the basic round-robin calculation, the calculation reduces the waiting time and turn-around time. Its implementation is divided into three parts, each of which provides optimal throughput.

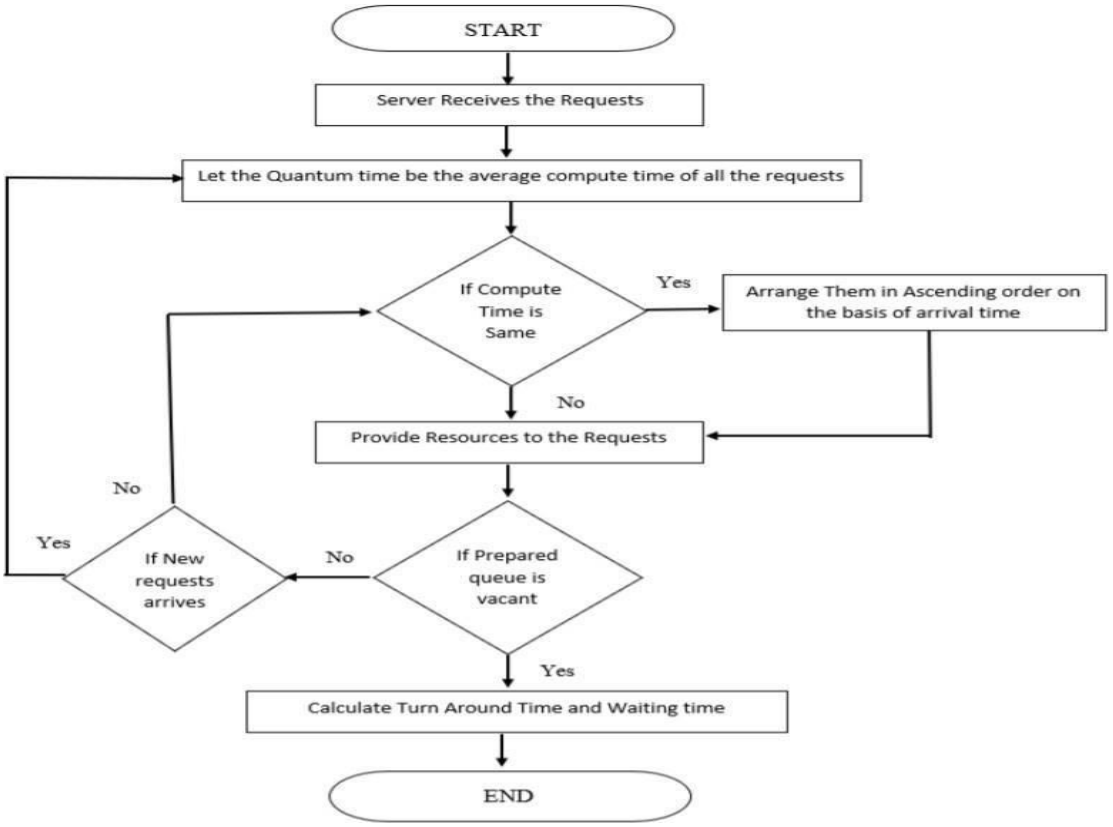
To overcome the drawbacks of fixed quantum, Mohamed et al. [8] proposed dynamic enhanced Round Robin algorithm Dynamic Priority Round Robin (DPRR) and Enhanced DPRR (EDPRR). Based on the processing time of jobs, the DPRR algorithm creates a dynamic time quantum. The second algorithm (EDPRR) selects jobs to be done based on their execution.

III. PROPOSED WORK

The proposed CPU Scheduling computation is based on a round-robin scheduling, in which the computation has been modified. When compared to the traditional round-robin scheduling algorithm, it drastically reduces the waiting time and turn-around time. Rather than providing a fixed quantum time during CPU computation, this technique determines quantum time on its own, basically focusing on a dynamic quantum time. Essentially, this paper dives into the concept in which the original RR scheduling is compared to one that has various modifications.

Initially, we preserve all of the requests in the server's arbitrary request queue as they arrive. The algorithm then calculates the mean of the burst time of a large number of requests at that point in the subsequent stage. It will define the quantum time effectively after determining the mean value. In the final stage, the algorithm chooses the first requests in the queue and allocates the resources/server for the duration of the mean quantum time for that request and then travels in a cyclic manner like the traditional round-robin, until a new request arrives, in which the is quantum time calculated again for the batch.

Flowchart For Proposed Optimos Algorithm:



The steps to be followed for the Proposed Algorithms are :

1. START
2. Load the prepared queue with the arriving requests
3. Compute the average of CPU burst time of the significant number of processes.
4. Assign the mean value as the quantum time for all the requests
5. Consider the burst time of all the requests in the queue, In the case of uniform burst time rearrange the requests on the basis of arrival time.
6. Grant CPU to the first request waiting in the ready queue and execute it for the computed quantum time.
7. Before executing the next process ensure the previous request is accomplished, if not then load previous request with the remaining burst time at the end of queue.
8. Before iterating again ensure if new requests has arrived,
    a. If yes, go to step 3 until the queue is vacant.
    b. If no, go to step 5 until the queue is vacant.
9. Calculate the average Waiting and turn around time
10. END.

IV. EXPERIMENTL ANALYSIS

For the purposes of evaluating the results of this algorithm, it is implicit to use an environment in which all calculations are performed on a single processor. Furthermore, all of the procedures have the same priority, which means they are arranged in the queue in the same order. The proposed work is written in the Java programming language. Various numbers of experiments are also completed, and their outcomes are considered before issuing final statements.

CASE 1: Here 5 processes are considered where the arrival time, burst time and the time quantum as specified prior to the execution.

Table 1: Process Specification

Time Quantum (TQ) = 15 ms		
Process	**Arrival Time(ms)**	**Burst Time(ms)**
1	0	35
2	0	20
3	0	45
4	0	10
5	0	65

Gantt Chart 1: Traditional Round Robin

P1	P2	P3	P4	P5	P1	P2	P3	P5	P1	P3	P5	P5	P5
15	30	45	55	70	85	90	105	120	125	140	165	180	185

Gantt Chart 2: Proposed Optimos Algorithm

P1	P2	P3	P4	P5	P3	P5
35	55	90	100	135	145	180

Table 2: Comparative Results

Process	Burst Time(ms)	Round Robin Algorithm (TQ = 15ms)		Proposed Optimos Algorithm (TQ = 35ms)	
		Waiting Time(ms)	Turn Around Time(ms)	Waiting Time(ms)	Turn Around Time(ms)
1	35	90	125	0	35
2	20	70	90	35	55
3	45	95	140	100	145
4	10	45	55	90	100
5	65	110	175	110	175

CASE 2: Here 5 processes are considered where the arrival time is different and the time quantum as specified prior to the execution.

Table 3: Process Specification

Time Quantum (TQ) = 50 ms		
Process	Arrival Time(ms)	Burst Time(ms)
1	0	18
2	0	66
3	0	26
4	12	102
5	35	75

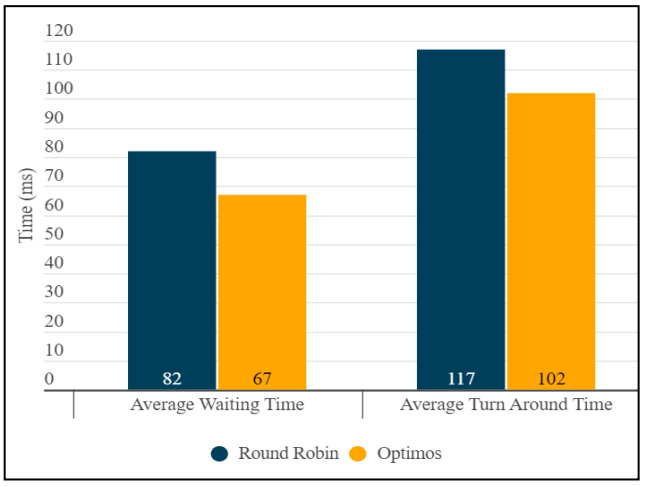
Table 4: Comparative Results

Process	Burst Time(ms)	Round Robin Algorithm (TQ = 50ms)		Proposed Optimos Algorithm (variable quantum time)	
		Waiting Time(ms)	Turn Around Time(ms)	Waiting Time(ms)	Turn Around Time(ms)
1	18	0	18	0	18
2	66	144	210	85	151
3	26	68	94	82	108
4	102	173	263	151	253
5	75	175	215	212	287

V. COMPARISON OF RESULTS

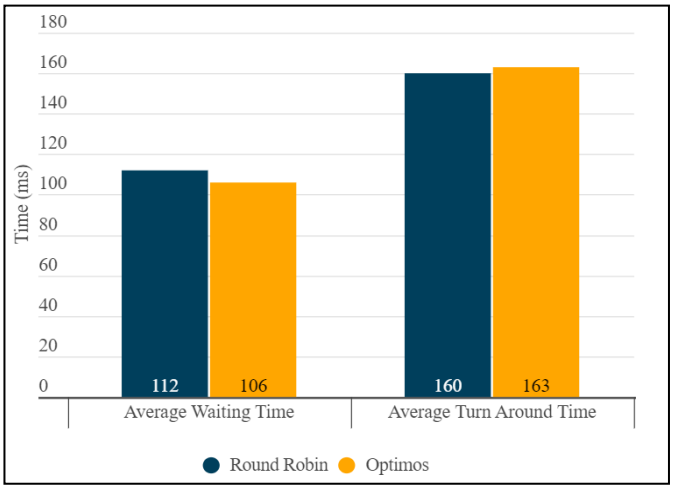
Performance Comparison for Case 1:

Performance Attribute	Original RR Scheduling	Proposed Model	Remarks
Average Waiting Time	82	67	15 units of time saved
Average Turn Around Time	117	102	15 units of time saved



Performance Comparison for Case 2:

Performance Attribute	Original RR Scheduling	Proposed Model	Remarks
Average Waiting Time	112	106	6 units of time saved
Average Turn Around Time	160	163	3 units of time Exceeded



VI. CONCLUSION AND FUTURE SCOPE

We compared the original round robin model to our proposed model and displayed the results. An improved version of round robin is proposed here. Based on the graphs and it was demonstrated through calculations that the best/worst case of the original algorithm is equal to the proposed algorithm's worst-case scenario. When compared to the cases of average waiting time and turn-around time, the graphs and results show that the proposed algorithm outperforms conventional round robin.

Round robin scheduling can be used in cloud computing to balance the load as soon as possible because response time is reduced effectively. Only the average waiting time and average turnaround time are compared in this report. The number of switch cases used in executing the algorithm has not been highlighted. In the future, we will improve the work using this attribute as well, and we will retrieve the results as they appear.

REFERENCES

[1] Dr.D.Ravindran, Ab Rashid Dar, Cloud Based Resource Management with Autoscaling, IJSRSET, Volume 2, Issue 4, 2016

[2] Hitoshi Matsumoto, Yutaka Ezaki," Dynamic Resource Management in Cloud Environment", July 2011, FUJITSU science & Tech journal, Volume 47, No: 3, page no: 270-276.

[3] Chandiramani, K.; Verma, R.; Sivagami, M. A Modified Priority Preemptive Algorithm for CPU Scheduling. Procedia Computer. Sci. 2019, 165, 363–369.

[4] Tajwar, M.M.; Pathan, N.; Hussaini, L.; Abubakar, A. CPU Scheduling with a Round Robin Algorithm Based on an Effective Time Slice. J. Inf. Process. Syst. 2017, 13, 941–950.

[5] V.A. Jane, B.J. Hubert Shanthan and L. Arockiam, A Survey of Algorithms for Scheduling in the Cloud: In a metric Perspective, International Journal of Computer Sciences and Engineering, Volume-6, Special Issue-2, PP 66 - 69, March 2018

[6] Vishnu Kumar Dhakad, Lokesh Sharma, " Performance Analysis of Round Robin Scheduling using Adaptive Approach based on Smart Time Slice and comparison with SRR", International Journal of Advances in Engineering & Technology, Vol. 3, Issue 2, pp. 333-339, May 2012, ISSN 2231-1963.

[7] Abdulraza, Abdulrahim, Salisu Aliyu, Ahmad M Mustapha, Saleh E Abdullahi, "An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm," International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 2, February 2014 ISSN: 2277 128X.

[8] B Mohamed, NE Al-Attar, W Awad, and FA Omara, Dynamic job scheduling algorithms based on round-robin for cloud environment, Res. J. Appl. Sci. Eng. Technol., Vol. 14, 2017, pp. 124-131.