Volume 2, No. 2, February 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

Design and Performance Evaluation of Multi Cyclic Round Robin (MCRR) Algorithm Using Dynamic Time Quantum

H. S. Behera[#1], Rakesh Mohanty[#2], Sabyasachi Sahu[#3], Sourav Kumar Bhoi[#4]

Department of Computer Science and Engineering,
Veer Surendra Sai University of Technology, Burla, Sambalpur, Orissa, India
[#1] hsbehera_india@yahoo.com,
[#2] rakesh.iitmphd@gmail.com
[#3] sabyasaschi.sahu1102@gmail.com,
[#4] souravbhoi@gmail.com

Abstract: The efficiency and performance of a system mainly depends on productive time and resource utilization. CPU scheduling algorithm gives a real time measurement of this productive utilization and its feasibility. CPU being considered a primary computer resource, its scheduling is central to operating-system design. A thorough performance evaluation of various scheduling algorithms indicates that Round Robin Algorithm is considered as optimal in time shared environment because the static time is equally shared among the processes. In this paper, we have proposed an improved scheduling algorithm by using dynamic time quantum and multi-cycled Round Robin concept. Our approach is based on the calculation of time quantum twice in a single round robin cycle. Experimental analysis shows that our proposed algorithm performs better than Round Robin algorithm. It also minimizes the overall number of context switches, average waiting time and average turn-around time.

Keywords : Scheduling, Round Robin, Context Switch, Waiting time, Turn around Time, Median, Upper Quartile

INTRODUCTION

Operating System is the interface between the hardware and the user (application program). It controls and coordinates the use of the hardware among various application programs for various users[3]. Modern operating systems have become more complex, they have evolved from a single task to a multitasking environment in which processes run in a concurrent manner[1]. In multitasking and multiprocessing environment the way the processes are assigned to run on the available CPUs is called *scheduling*. The main goal of the scheduling is to maximize the different performance metrics viz. CPU utilization, throughput and to minimize response time, waiting time and turnaround time and the number of context switches[2]. Scheduling is often implemented in diverse real time applications like routing of data packets in computer networking, controlling traffic in airways, roadways and railways, scheduling of league games etc. This assignment is carried out by software known as a *scheduler* and/or *dispatcher.* Operating systems may feature up to 3 distinct types of a *long-term scheduler* a *mid-term or medium-term scheduler* and a *short-term scheduler*. The names suggest the relative frequency with which these functions are performed. In Round Robin (RR) every process has equal priority and is given a time quantum or time slice after which the process is preempted. Although RR gives improved response time and uses shared resources efficiently. Its limitations are larger waiting time, undesirable overhead and larger turnaround time for processes with variable CPU bursts due to use of static time quantum This motivates us to implement RR algorithm with sorted remaining burst time with dynamic time quantum concept. Another concept employed in this algorithm is the use of more than one cycle instead of a single Round Robin.

A. Preliminaries

A program in execution is called a *process*. The processes, waiting to be assigned to a processor are put in a data structure entity called *ready queue*. The time for which a process holds the CPU is known as *burst time*. The time at which a process arrives for execution is its *arrival time*. *Turnaround time* is the amount of time to execute a particular process, while *waiting time* is the amount of time a process has been waiting in the ready queue. Time expired from the submission of a request by the process till its first response is defined as the *response time*. *Scheduler* selects a process from queues in a manner, for its execution such that the load balance is effective. In *non-preemption,* CPU is assigned to a process; it holds the CPU till its execution is completed. But in *preemption*, running process is forced to release the CPU by the newly arrived process. In *time sharing system*, the CPU executes multiple processes by switching among them very fast. The number of times CPU switches from one process to another is called as the number of *context switches.*

B. Scheduling algorithms

When there are number of processes in the ready queue, the algorithm which decides the order of execution of those processes is called a *scheduling algorithm.* Various well known CPU scheduling algorithms have been developed viz. First Come First Serve algorithm (FCFS), Shortest Job First algorithm (SJF) and Priority scheduling algorithm. All the above algorithms are non-preemptive in nature and are not suitable for time sharing systems. Shortest Remaining Time

Next (SRTN) and Round Robin (RR) are preemptive in nature. RR is most suitable for time sharing systems. But its average output parameters (waiting time, turn-around time etc. are not feasible enough to be employed in real-time systems.

C. Related Work

Abielmona[1] on account of his analytical scrutiny of a innumerable number of scheduling algorithms gives a thorough insight into the factors affecting the performance parameters of a scheduling algorithm. RR algorithm gives better responsiveness but worse average turn-around and waiting time. The Proportional Share Scheduling Algorithm proposed by Helmy and Dekdouk[2] combines low overhead of round robin algorithms besides favoring shortest jobs. Weight readjustment in the algorithm enables existing proportional share schedulers to significantly reduce, the unfairness in their allocation. The static time quantum which is a limitation of RR was removed by taking dynamic time quantum by Matarneh[5]. A rule of thumb is also stated that 80% of the CPU bursts should be shorter than the time quantum[3]. Recently improved variants of round robin algorithms SRBRR[4] and PBDRR[6] have been developed. The time quantum that was repeatedly adjusted on a run-time basis according to the burst time of the running processes are considered to improve the waiting time, turn-around time and number of context switches.

E. Organization of the paper

The paper is divided into four sections. Section I gives a brief introduction on the various aspects of the scheduling algorithms, the approach to the current paper and the motivational factors leading to this improvement. Section II presents the materials and methods used, the pseudo code and illustration of our proposed new algorithm (MCRR). In section III, an experimental analysis and Result of our algorithm (MCRR) and its comparison with the static RR algorithm and dynamic SRBRR algorithm is presented. Conclusion is presented in section IV followed up by the references used. Tables and figures used have been represented by numbers.

MATERIALS AND METHODS

1. Sort the processes in ascending order of their burst times
   n → number of processes
   i, → counter value = 0
While (ready queue is ! = NULL)
2. M= medianth process
   MTQ = median(burst time of all processes in ready queue)
   UTQ= UpperQuartile(n-m)(burst time of all remaining processes in ready queue after calculating the median)
3. If (i< m)
//Assign CPU to process P_i and give it time of slice= MTQ.
   $P_i \rightarrow MTQ_{cpu}$
   i++;
  else
//Assign CPU to process P_i and give it time of slice= UTQ.
   Assign UTQ to process
   $P_i \rightarrow UTQ_{cpu}$
4. If(i < n)
   Goto step 2
   i++
5. If new process is arrived
   update the counter n and go to step 1.
  End of While.
6. Average waiting time, average turnaround time and number of context switches and CRITERIA % are calculated.
7. End

Fig 1: Pseudo code for Multi Cyclic Round Robin algorithm

In our work, the RR algorithm is improvised by an astute distribution of time quantum of processes, repeatedly over the whole Round Robin cycle. Static time quantum being a limitation of RR algorithm, we have used the concept of dynamic time quantum. Besides, we have supplemented the use of median with upper quartile, as two concepts in one cycle of RR. Implicating that up to the medianth process, we use time quantum MTQ calculated by Median Quartile formula and for the succeeding processes, we use the Upper Quartile formula to calculate the time quantum UTQ. This time quantum is used by the remaining processes and this continues up to the execution of all the processes. In succeeding cycles of the round robin, the median and upper quartiles are again calculated taking into consideration the remaining processes.

Formula1 represents the calculation of time quantum by Median Quartile MQ:

$$MQ = \begin{cases} Y_{(N+1)/2} & \text{if N is odd} \\ \frac{1}{2}(Y_{N/2}) + (Y_{1+N/2}) & \text{if N is even} \end{cases}$$

where, Y is the number located in the middle of the group of numbers in ascending order and N is the number of processes. Formula 2 represents the calculation of time quantum by Upper Quartile Q3:

$UQ = \frac{3}{4}(N+1)$ where N is the number of processes

So by these formulas we calculate the time quantum in our proposed algorithm.

CRITERIA= [{MTQ*m} + {UTQ*(N-m)}]/N
This variable is used for comparison with the 80% criterion.

PROPOSED MCRR ALGORITHM

In our proposed algorithm, the time quantum is taken as the burst time of the median of all the processes. The scheduling continues with the same time quanta up to the medianth process. For the succeeding process the time quantum is determined by taking the burst time of Upper Quartile of all processes. This whole operation occurs in a single scheduling cycle of the processes sorted in ascending order of the burst time of all the processes.

A. Uniqueness of our Approach

In our algorithm, the jobs are sorted in ascending order of their burst time to give better turnaround time and waiting time like SRTN Algorithm. Performance of RR algorithm solely depends upon the size of time quantum. If it is very small, it causes too many context switches. If it is very large, the algorithm degenerates to FCFS. Taking into account the "80% criteria", our algorithm employs the use of dynamic quantum and also multi calculation of time quantum in a single cycle of round robin.

B. Pseudo code of Proposed Algorithm

In our algorithm, when processes are already present in the ready queue, their arrival times are assigned to zero before they are allocated to the CPU. The burst time and the number of processes (n) are accepted as input. Let TQ be the time quantum. i and other integers specified are either counters or flag bits.

C. Illustration

Given the burst time sequence: 54 99 5 27 32. Initially the burst time of all the processes were sorted in ascending order which resulted in sequence 5 27 32 54 99. Then the *median* of the above burst time which was calculated to be 32 (MTQ) was assigned as the time quantum up to the median position of the processes. In the next step burst time for the rest processes are calculated by applying *upper quartile* method and it is found to be 99 (UTQ) only because after the last process no other process are there. When a process completes its burst time, it gets deleted from the ready queue automatically. So in this case, the processes P1, P2 and P3 were deleted from the ready queue, then P3 and P5 was given 99 as the time quantum so that it completes its execution. If the average time quantum is calculated using the MTQ and UTQ which is calculated by using the CRITERIA percentage, we find that the algorithm is close to the 80% criteria. The above process was continued till all the processes were deleted from the ready queue.

RESULTS

A. Assumptions

The environment where all the experiments are performed is a single processor environment and all the processes are independent. Time slice is assumed to be not more than the maximum burst time. All the attributes like burst time, number of processes and the time slice of all the processes are known before submitting the processes to the processor. All processes are CPU bound. No processes are I/O bound. Also, a large number of processes is assumed in the ready queue for better efficiency.

B. Experimental Frame Work

Our experiment consists of several input and output parameters. The input parameters consist of burst time, arrival time, time quantum and the number of processes. The output parameters consist of average waiting time, average turnaround time and number of context switches.

C. Data set

We have performed six experiments for evaluating performance of our new proposed algorithm. For the first three experiments, we have considered the data set as the processes with burst time in increasing, decreasing and random order respectively. In the above three cases, the arrival time was assumed to be the same. Then the above experiments were performed by considering data set with different arrival time for each process.

D. Performance Metrics

The significance of our performance metrics for our experiment is as follows.:
1) Turnaround time (TAT): For the better performance of the algorithm, average turnaround time should be less.
2) Waiting time (WT): For the better performance of the algorithm, average waiting time should be less.
3) Number of Context Switches (CS): For the better performance of the algorithm, algorithm, the number of context switches should be less.

E. Experiments Performed:

To evaluate the performance of our proposed algorithm, we have taken a set of five processes in six different cases. For simplicity, we have taken 5 processes. The algorithm works effectively for a very large number of processes. In each case, we have compared the experimental results of our proposed algorithm with the round robin scheduling algorithm with fixed time quantum Q, SRBRR with dynamic time quantum TQ and MCRR with dynamic time quanta MTQ and UTQ. Here we have assumed a constant time quantum Q equal to 25 in all the cases for RR, dynamic time quantum TQ and MTQ calculated by the median formula and the second dynamic time quantum UTQ calculated by the Upper Quartile formula.

Case 1: We assume five processes arriving at arrival time= 0, with increasing burst time (P1 = 13, P2 = 35, P3 = 46, P4 = 63, p5= 97) as shown in TABLE-1(upper). The Table-1(lower) shows the output using RR algorithm SRBRR algorithm and our new proposed algorithm.Figure-2 and Figure-3 and Figure 4 shows Gantt chart for both the algorithms respectively.

Processes	Arrival Time	Burst Time
P1	0	13
P2	0	35
P3	0	46
P4	0	63
P5	0	97

Algorithm	Time Quantum	Average TAT	Average WT	CS
RR	25	148.2	97.4	11
SRBRR	46,34,17	122.4	71.6	7
MCRR	46,97	113.2	62.4	4

Table 1: Comparison between RR algorithm SRBRR algorithm and MCRR (case 1).

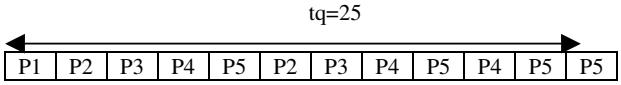

Fig.2: Gantt chart for RR in Table 1 (case1)

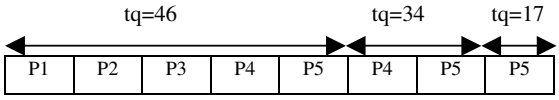

Fig.3: Gantt chart for SRBRR in Table 1(case 1)

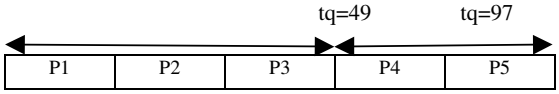

Fig.4: Gantt chart for MCRR in Table 1(case 1)

Case 2: We assume five processes arriving at arrival time =0, with decreasing burst time (P1 = 86, P2 =53, P3 = 32, P 4= 21, p5= 9) as shown in Table-2(upper). The Table-2(lower) shows the output using RR algorithm, SRBRR algorithm and our new proposed algorithm. Figure-6 and Figure-7 and Figure-8 shows Gantt chart for both the algorithms respectively.

Processes	Arrival Time	Burst Time
P1	0	86
P2	0	53
P3	0	32
P4	0	21
P5	0	9

Algorithm	Time Quantum	Average TAT	Average WT	CS
RR	25	150.8	110.5	10
SRBRR	32,38,16	89.8	49.6	7
MCRR	32,86	83.4	43.2	4

Table 2: Comparison between RR and MCRR (case 2)

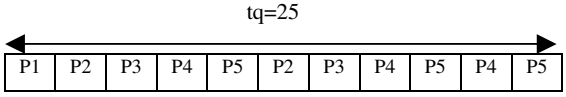

Fig.6: Gantt chart for RR in Table 2(case 2)

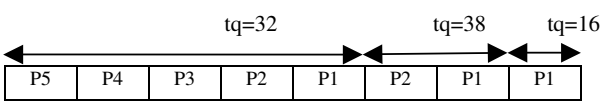

Fig.7: Gantt chart for SRBRR in Table 2(case 2)

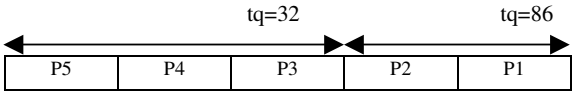

Fig.8:Gantt chart for MCRR in Table 2 (case 2)

Case 3: We Assume five processes arriving at arrival time = 0, with random burst time (P1 = 54, P2 = 99, P3 = 5, P4 =27, p5= 32) as shown in TABLE-3 (upper). The Table-3(lower) shows the output using RR algorithm, SRBRR algorithm and our new proposed algorithm. Figure-10, Figure-11and Figure-12 shows Gantt chart for both the algorithms respectively.

Processes	Arrival Time	Burst Time
P1	0	54
P2	0	99
P3	0	5
P4	0	27
P5	0	32

Algorithm	Time Quantum	Average TAT	Average WT	CS
RR	25	152.2	108.8	11
SRBRR	32,45,22	93.6	50.2	7
MCRR	32,99	87.2	43.5	4

Table 3: Comparison between RR and MCRR (case 3)

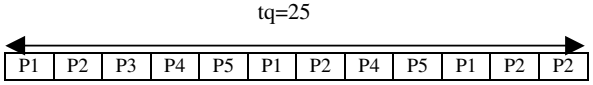

Fig.10: Gantt chart for RR in Table 3 (case 3)

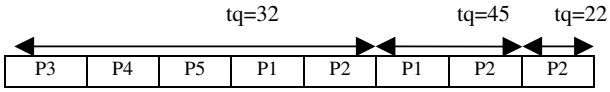

Fig.11: Gantt chart for SRBRR in Table 3 (case 3)

Fig.12: Gantt chart for MCRR in Table 3 (case 3)

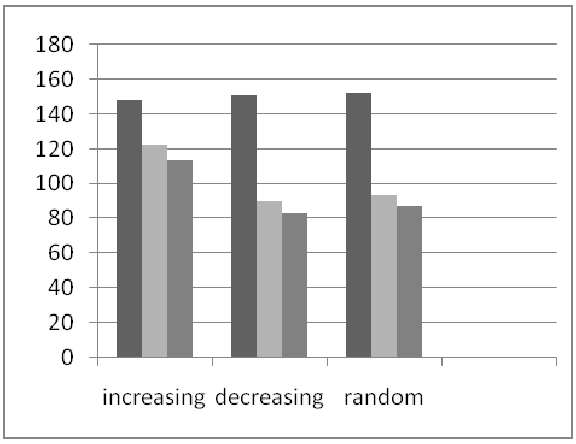

Fig.13: Comparison of average turnaround time of RR, SRBRR and MCRR taking static and dynamic time quantum for increasing, decreasing and random burst sequence.

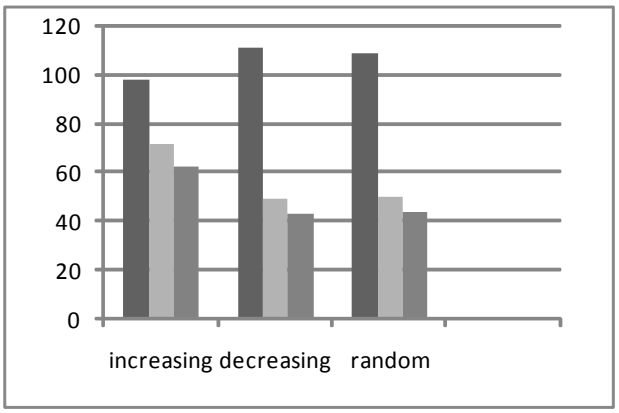

Fig.14: Comparison of average waiting time of RR, SRBRR and MCRR taking static and dynamic time quantum for increasing, decreasing and random burst sequence.

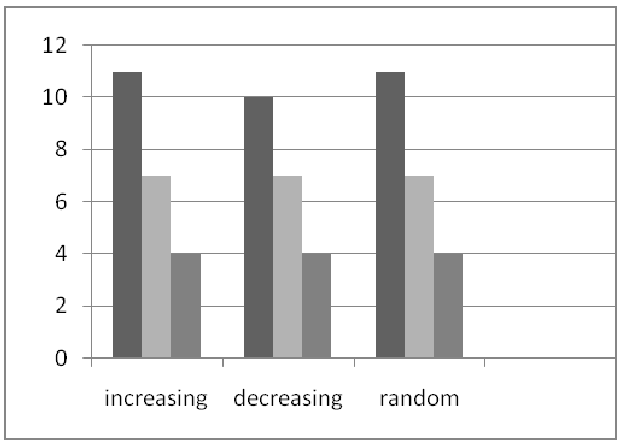

Fig.15: Comparison of context switch of RR, SRBRR and MCCR taking static and dynamic time quantum for increasing, decreasing and random burst sequence.

DISCUSSION AND CONCLUSION

Quite a few measures have been proposed for bettering the output parameters of the scheduling algorithms viz. the turn-around time, waiting time and the overhead of context switches in round robin mode. Even so, all methodologies employed calculate time quantum only once in a particular Round robin cycle. Our proposed algorithm employs a line of attack in which the time quantum is calculated twice viz. MTQ and UTQ in a single Round Robin cycle. This approach optimizes the critical performance metrics in a simple yet effective manner. Simulation of this algorithm upon the data sets clearly shows an inclusive improvisation of all performance parameters. The algorithm needs a plethora of processes in the ready queue for it to work effectively. Otherwise it just reduces to a preemptive FCFS. The algorithm is close to fulfillment of the 80% criteria. Hence the concept of multiple calculation of time quantum in a single round robin cycle has proved effective in designing an operating system that is close to an ideal scheduling of processes. Time quantum is the performance driving parameter in a scheduling algorithm. So the choice of time quantum in a scheduling algorithm must be precise. The current research and the experimental analysis substantiates the use of dynamic rime quantum and multi-cycle round robin for an effective scheduling.

REFERENCES

[1] Rami Abielmona, Scheduling Algorithmic Research, Department of Electrical and Computer Engineering Ottawa-Carleton Institute, 2000.

[2] Tarek Helmy, Abdelkader Dekdouk, "Burst Round Robin: As a Proportional-Share Scheduling Algorithm." IEEEGCC, King Fahed University.

[3] Silberschatz, A., P.B. Galvin and G.Gagne, Operating Systems Concepts.7th Edn., John Wiley and Sons, USA ISBN: 13:978- 0471694663, pp: 944, 2004.

[4] Rakesh Mohanty, H.S. Behera and et. al, Design and Performance Evaluation of a new proposed Shortest Remaining Burst Round Robin(SRBRR) scheduling algorithm, Proceedings of the International Symposium on Computer Engineering and Technology(ISCET), March, 2010.

[5] Rami J. Matarneh , Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the now Running Processes, Department of Management Information Systems, American Journal of Applied Sciences 6 (10):1831-1837, 2009,.

[6] Rakesh Mohanty, H.S. Behera and et. al., Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems, International Journal of Advanced Computer Science and Applications(IJACSA), Vol 2 No. 2, pp 46-50, February 2011.

Volume 2, No. 2, February 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

SHORT BIODATA OF ALL THE AUTHOR

1. Prof. H.S. Behera is currently working as a faculty in Dept. of Computer Science and Engineering, VSS University of Technology, Burla, Orissa, India. His research areas of interest include Operating Systems, Data Mining and Distributed Systems.

2. Prof. Rakesh Mohanty is currently working as a faculty in Dept. of Computer Science and Engineering, VSS University of Technology, Burla, Orissa, India. His research areas of interest include Operating Systems, Data Structures and Algorithms.

3. Sabyasachi Sahu is a 4th year undergraduate B. Tech. student in Dept. of Computer Science and Engineering, VSS University of Technology, Burla, Orissa, India.

4. Sourav Kumar Bhoi is a 4th year undergraduate B. Tech. student in Dept. of Computer Science and Engineering, VSS University of Technology, Burla, Orissa, India.