Global Journal of Computational Intelligence Research.
ISSN 2249-0000 Volume 1, Number 2 (2011), pp. 149-158

http://www.ripublication.com/gjcir.htm

A Novel Round Robin CPU Scheduling using Shortest Burst Approach based on Smart Time Slice

[1]Saroj Hiranwal, [2]Vishnu Kumar Dhakad, [3]K.C. Roy and [4]Dharm Singh

[1]Dept. of CSE, Suresh Gyan Vihar University
Jaipur, Rajasthan, India
[2]Dept. of CSE, SBCET, Jaipur, Rajasthan, India
[3]Dept. of ECE, Pacific University, Udaipur, Rajasthan, India
[4]Dept of CSE, MPUAT, Udaipur, Rajasthan, India
E-mail: ersaroj_hiranwal@rediffmail.com, vishnudhakad@rediffmail.com, roy.krishna@rediffmail.com, dharm@mpuat.ac.in

Abstract

In this paper, we have studied various algorithms for scheduling of the processes in Operating System (OS).Amongst all the algorithms, Round Robin (RR) performs optimally. The Round Robin Scheduling has disadvantage is longer average waiting time, higher context switches, higher turnaround time and low throughput. To improve the performance of RRT the new proposed algorithm called "Novel Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice". It is a Priority Driven Scheduling algorithm based on burst time of processes. First of all we arrange the processes according to the execution time/burst time in increasing order that is smallest the burst time higher the priority of the running process. The next idea of this approach is to choose the smart time slice (STS) is mainly depends on number of processes. The smart time slice is equal to the mid process burst time of all CPU burst time when number of process given odd. If number of process given even then we choose the time quantum according to the average CPU burst of all running processes. Based on the experiments and calculations the proposed algorithm radically solves the fixed time quantum problem which is considered a challenge for Round Robin Scheduling Algorithm. The use of scheduling algorithm increased the performance and stability of the operating system. Empirical results are presented to compare performance with previously proposed algorithms

Keywords: CPU, STS, FCFS, RR, SJF, FIFO, PCB

Introduction

An operating system interacts between the user and the computer hardware. The purpose of an operating system is to provide a platform in which a user can execute programs in well-located and efficient manner. Modern operating systems and time sharing systems are more complex, they have evolved from a single task to a multitasking environment in which processes run in a synchronized manner. CPU scheduling is a necessary operating system task; therefore its scheduling is central to operating system design. When there is more than one process in the ready queue or job pool waiting its turn to be assigned to the CPU, the operating system must decide through the scheduler the order of execution. 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, response time and average waiting time for processes and the number of context switches.

CPU scheduling is the mechanism by which a resource is allocated to a process or task and executes in different ways. For scheduling many scheduling algorithms are used like FCFS, SJF, RR, and Priority scheduling algorithm. The processes are scheduled according to the given burst time, arrival time and priority. The execution of processes used number of resources such as Memory, CPU etc. A scheduling decision refers to the concept of selecting the next process for execution. During each scheduling decision, a context switch occurs, meaning that the current process will stop its execution and put back to the ready queue and another process will be dispatched. We define the scheduling overhead cost when more context switches and all process are switching the finally CPU performance will be decreased.

Scheduling algorithms are widely used in communications networks and in operating systems to allocate resources to competing tasks. In operating systems, the scheduler must order the set of running processes for access to the CPU and other system resources. This research investigates several well known CPU scheduling algorithms by means of analysis and compares their performance under different workloads. The aim of process scheduling is to assign the processor or processors to the set of processes in a way that meets system and user objectives such as response time, throughput or processor efficiency. Various scheduling algorithms have been proposed; many are well understood. This paper focuses to enable a quantitative comparison of the performance of the Round Robin algorithm under a range of workloads. Schedulers can be either work conserving or non-work conserving. Work conserving schedulers always schedule a task if one is run-able, whereas non-work conserving schedulers may idle the CPU even if there is a run-able task, typically to meet some performance constraint for the application, such as ensuring that it runs with a stringent periodicity.

Round Robin Algorithm

In this algorithm process are scheduled according to first come first serve manner and we assume fixed time quantum on random basis, if the time quantum is small than more context switches and more overhead and if large than it becomes FCFS Scheduling

Novel Round Robin Algorithm

The Novel Round Robin Scheduling Algorithm focuses on the drawbacks of simple Round Robin Algorithm which gives equal portion of time to all the processes (processes are scheduled in first come first serve manner) because of all the drawbacks in Round Robin Algorithm is not efficient for processes with smaller CPU burst. This result leads to the increase in waiting time and response time of processes which decrease the system throughput. The proposed algorithm eliminates the drawbacks of implementing a simple round robin algorithm in by scheduling of processes based on the CPU execution time. The allocated processor used to reduce the burden of the main processor is assigned processes according to the priority basis; the smaller CPU burst of the process, higher the priority. The proposed algorithm solves the problem of higher average waiting time, turnaround time, response time and more context switches thereby improving the system performance.

Intelligent Time Slice for Novel Round Robin

The proposed algorithm eliminates the defects of implementing simple Round Robin (RR) architecture in operating system by introducing a concept called smart time slicing which depends on three aspects they are priority, average CPU burst or mid process CPU burst, and context switch avoidance time. The proposed algorithm allows the user is to assign priority to the system based on execution time or burst time. The calculated smart time slice will be based on all CPU burst of new running processes. The smart time slice calculated according to the processes burst time; if the number of process are assigned into the ready queue are odd the smart time slice will be the mid process burst time else the number of processes are even in ready queue the smart time slice is average of all processes burst time is assigned to the processes.

Smart Time Slice = Mid Process Burst Time (If number of processes are odd)

Or

Smart Time Slice = Average Burst Time (If number of processes are even)

Then processes are executing according to the smart time slice and give superior result comparison to existing simple Round Robin Scheduling Algorithm and can be implemented in operating system.

Novel Round Robin Pseudo code

1. First of all check ready queue is empty
2. When ready queue is empty then all the processes are assigned into the ready queue.
3. All the processes are rearranged in increasing order that means smaller burst time process get higher priority and larger burst time process get lower priority.
4. While (ready queue != NULL)
5. Calculate smart Time Slice:

If (Number of process%2= = 0)

STS = average CPU burst time of all processes
Else
STS = mid process burst time

6. Assign smart time slice to the i^{th} process:
   Pi STS

7. If (i< Number of process) then go to step 6.
8. If a new process is arrived update the ready queue, go to step 2.
9. End of While
10. Calculate average waiting time, turnaround time, context switches and throughput.
11. End

Experimental Analysis

Formulation of Research Hypothesis with Expected Outcomes

To evaluate the performance of my proposed algorithm, we have taken a set of processes in different cases. Here for simplicity, we have taken 5 or 4 processes. The algorithm works effectively even if it used with a very large number of processes. In each case, we have compared the experimental results of my proposed algorithm with the Simple Round Robin Scheduling Algorithm with fixed time quantum Q. Here we have assumed a constant time quantum Q for simple RR and compare with my result. In my calculation I have varied the smart time slice depend on the number of processes. The smart time slice can be calculated according to proposed plan.

Table 4.12: Comparison of simple RR and Novel RR.

Algorithm	Time Quantum	CS	Average WT	Average TAT	Throughput
Simple RR	16	6	44.78	64.7	Low
SRBRR	20,12	3	21.25	41.25	High
Novel RR	20	3	21.25	41.25	High

Performance Charts With Respect To Number of Processes

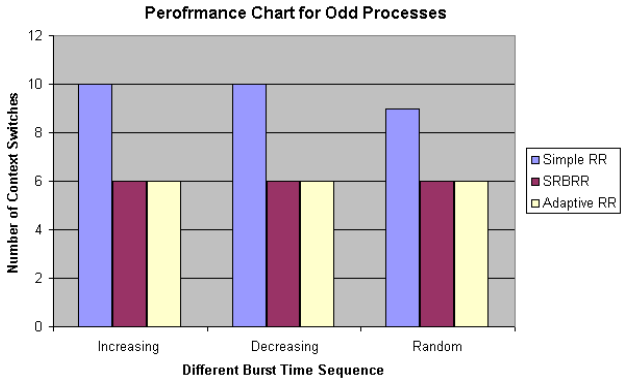


Figure 5.1: Comparison of Context Switches in order to increasing, decreasing and random burst time of odd number of processes.

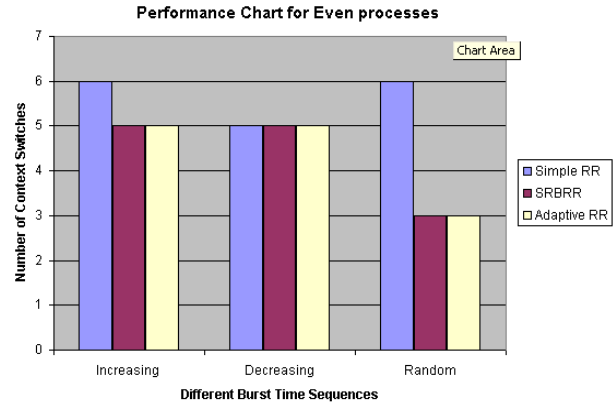


Figure 5.2: Comparison of Context Switches in order to increasing, decreasing and random burst time of even number of processes.

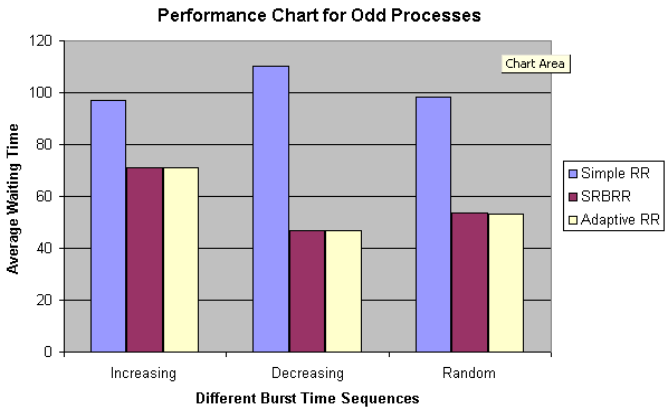


Figure 5.3: Comparison of average waiting time in order to increasing, decreasing and random burst time of odd number of processes.

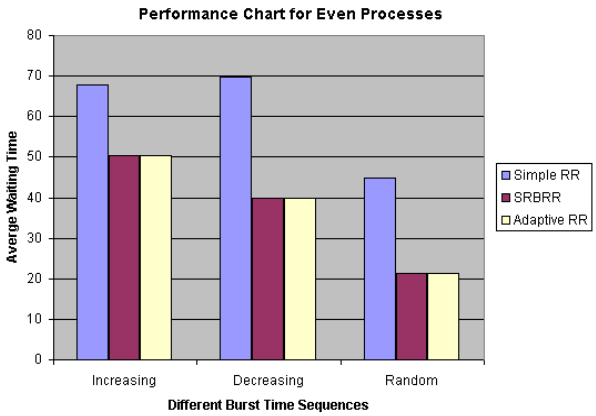


Figure 5.4: Comparison of average waiting time in order to increasing, decreasing and random burst time of even number of processes.

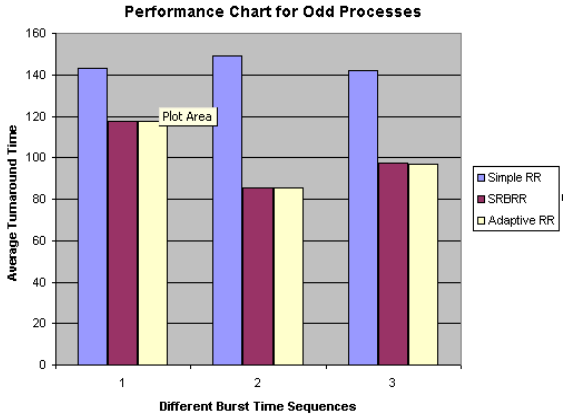


Figure 5.5: Comparison of turnaround time in order to increasing, decreasing and random burst time of odd number of processes.

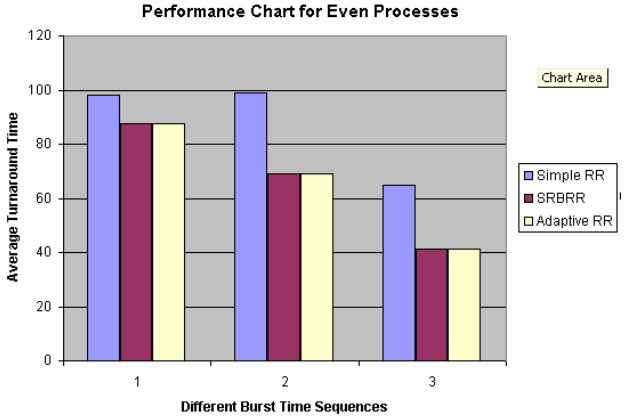
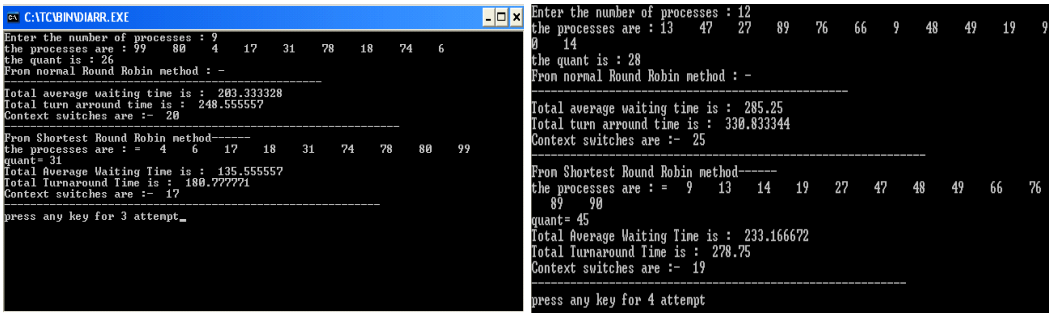
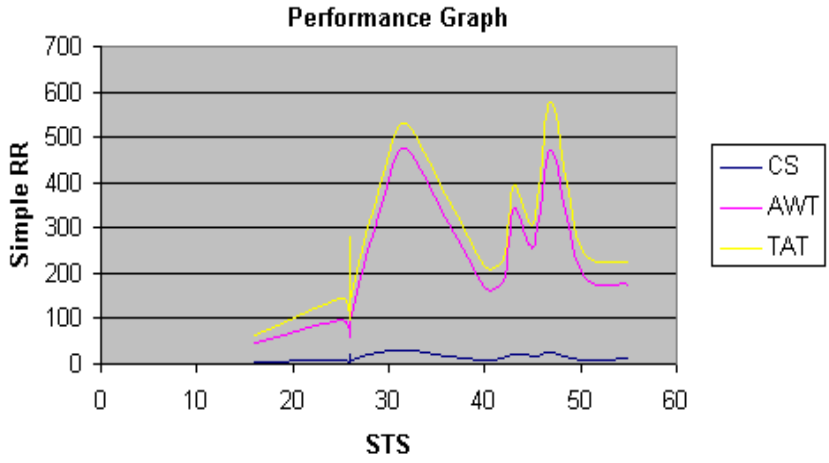


Figure 5.6: Comparison of turnaround time in order to increasing, decreasing and random burst time of even number of processes.

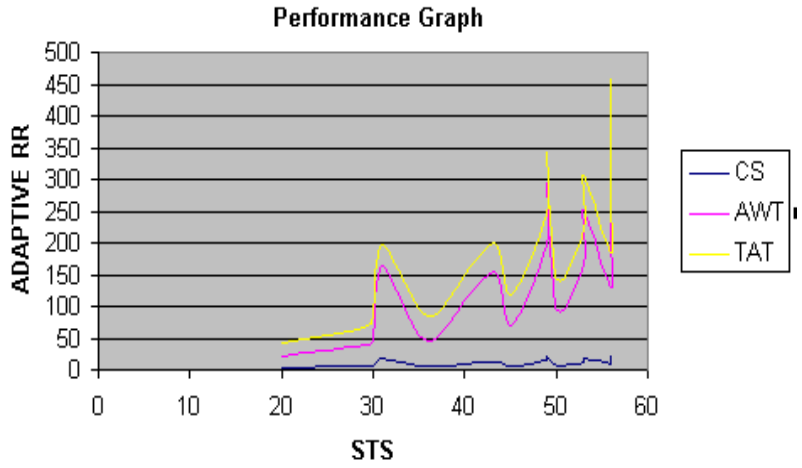
Decreasing and random burst time of even number of processes using Simple RR and Novel RR.



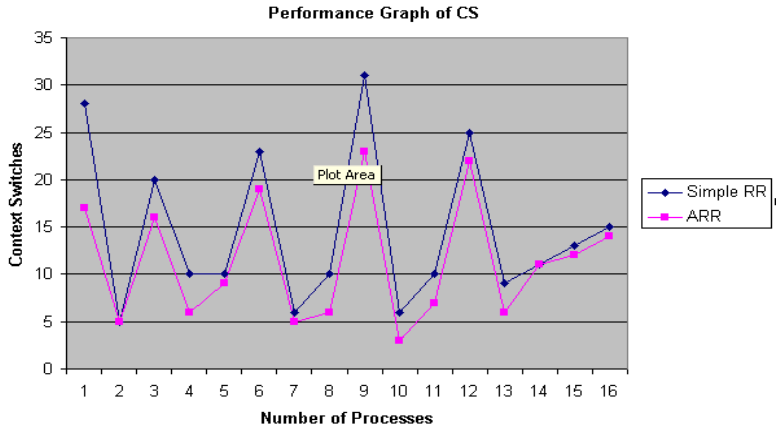
Performance graph of Simple RR



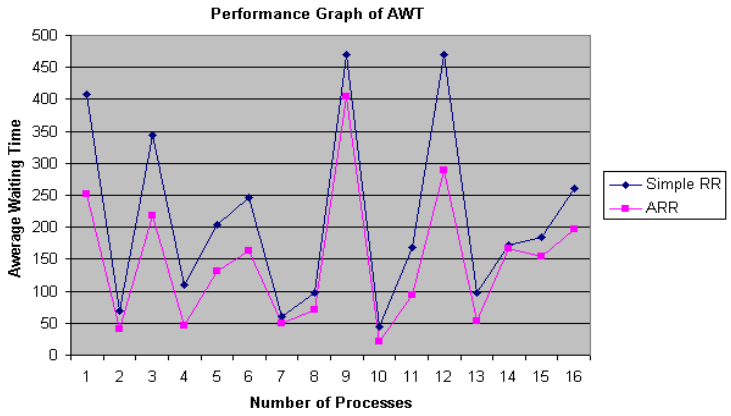
Performance graph of Novel RR



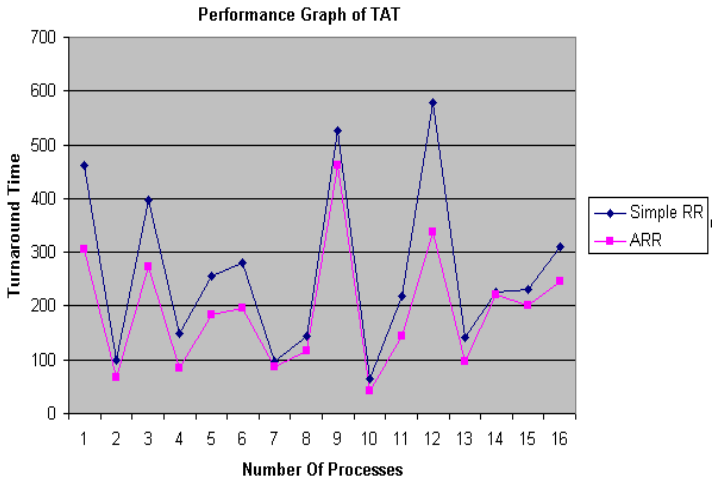
Performance graph for CS between Simple RR Vs Novel RR



Performance graph for AWT between Simple RR Vs Novel RR



Performance graph for TAT between Simple RR Vs Novel RR



Conclusion

The Novel Round Robin Algorithm is designed to meet all scheduling criteria such as maximum CPU utilization, minimum average waiting time, minimum average turnaround time and fewer contexts switches. According to the result of this algorithm all the CPU scheduling criteria is perfect found to the comparison of Simple Round Robin Scheduling Algorithm. The Novel RR algorithm uses fixed time quantum for computing the new average waiting time, average turnaround time, context switches and high throughput. The throughput is inversely proportional to context switches. If number of context switches is low then throughput will be high. This implementation based on burst time with smart time slice. After some mathematical calculation the proposed algorithm gives the better result in comparison of Simple Round Robin Scheduling Algorithm. And found that it produces lower average waiting time, average turnaround time, lower context switches. It is recommended to use the shortest burst time concept with smart time slice; because it will give the operating system the ability to adapt to the user behavior and not vice versa, which may lead us to rethink building an intelligent, learnable and adaptable operating system.

References

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

[2] "Tanebaun, A.S., 2008" Modern Operating Systems. 3rd Edn., Prentice Hall, ISBN: 13: 9780136006633, pp: 1104.

[3] "Tarek Helmy, Abdelkader Dekdouk" Burst Round Robin: As a Proportional-Share Scheduling Algorithm, IEEE Proceedings of the fourth IEEE-GCC Conference on towards Techno-Industrial Innovations, pp. 424-428, 11-14 November,2007.

[4] "Yaashuwanth .C & R. Ramesh" Inteligent time slice for round robin in real time operating system, IJRRAS 2 (2), February 2010.

[5] "Prof. Rakesh Mohanty, Prof. H. S. Behera, Khusbu Patwari, Manas Ranjan Das, Monisha Dash, Sudhashree" Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm, Am. J. Applied Sci., 6 (10): 1831-1837, 2009.

[6] "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, ISSN 1546-9239.

[7] "Shreedhar, M.; Varghese,G. (October 1995)" Efficient fair queueing using deficit round robin .ACM SIGCOMM Computer Communication Review 25 (4): 231. Doi:10.1145/21739.

[8] "Tong, W. and J. Zhao, 2007" Quantum varying deficit round robin scheduling over priority queues. Proceedings of the International Conference on

Computational Intelligence and Security, Dec. 15- 19, Computer Society, Harbin, China, pp: 252-256.

[9] "Back, D.S., K. Pyun, S.M. Lee, J. Cho and N. Kim, 2007" A hierarchical deficit round-robin scheduling algorithm for a high level of fair service. Proceedings of the International Symposium on Information Technology Convergence, Nov. 23-24, IEEE Computer Society, Washington DC., USA., pp: 115-119.