International Journal of Computer Science and Communication Vol. 2, No. 2, July-December 2011, pp. 319-323

Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice

SAROJ HIRANWAL, Dr. K.C. ROY
[1] Department of Computer Science and Engineering, Suresh Gyan Vihar University Jaipur, Rajasthan, India
[2] Department of Electronics and communication Engineering, Pacific University Udaipur, Rajasthan, India

ersaroj_hiranwal@rediffmail.com, roy.krishna@rediffmail.com

Abstract: The performance and ability of multitasking and time sharing system is becomes more multifaceted and calculating the averaging waiting time, turnaround time, response time, and context switches from the number of processes mainly depends on the used CPU scheduling algorithm where the CPU is one of the most important computer resource and as Round Robin scheduling is considered most widely used scheduling algorithms. The Round Robin Scheduling has disadvantage is longer average waiting time, higher context switches, higher turnaround time and low throughput. In Round Robin Scheduling the time quantum play a very important role for scheduling, because if time quantum is very large then Round Robin Scheduling Algorithm is same as the FCFS Scheduling. If the time quantum is extremely too small then Round Robin Scheduling is called as Processor Sharing Algorithm and number of context switches is very high. In this research a new proposed algorithm is presented which is discussed in detail, tested and verified. After that I reduce average waiting time, context switches, turnaround time, and throughput of simple Round Robin Scheduling by using a new scheduling algorithm. The new proposed algorithm called "Adaptive 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 and support building of an self-adaptation operating system, which means that the system is who will adapt itself to the requirements of the user and not vice versa.
Keywords: CPU, STS, FCFS, RR, SJF, FIFO, PCB

1. 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.

Scheduling algorithms are the mechanism by which a resource is allocated to a process or task. 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.

2. PRELIMINARIES

2.1 Definitions

Program is refers to the set of instructions that are executed in pipeline fashion. Program in execution is called *process*. In operating system each process is represented by a *process control block*(PCB).The PCB contain many information about the process such as process state, process number, program counter, list of open files, registers and CPU scheduling information. When processes enter the system they are put into a job pool. This job pool refers to the *job queue*. The processes that are residing in main memory and are ready and waiting to execute are kept on a list is called *ready queue*. *Device queue* refers to the when a process assign to the CPU, it executes or while waiting for some I/O devices for completing particular I/O event. So, the list of processes waiting in queue for particular I/O devices is called a device queue. The *long term scheduler* is also known as job scheduler that means it selects the process from the job pool and loads into the memory for execution. The *short term scheduler* is known as CPU scheduler that means it selects the processes that are ready to execute and allocates the CPU to one of them. The *medium term scheduler* is used in time sharing system. The key idea behind a medium term scheduler is that some times it can be advantageous to remove processes from memory and thus reduce degree of multiprogramming. Later, the processes 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. The time for which a process holds the CPU is known as burst time. The time at which a process arrives is its *arrival time*. *Turnaround time* is the amount of time to execute a particular process. *Waiting time* is the amount of time a process has been waiting in the ready queue. Time from the submission of a request by the process till its first response is the *response time*.

2.2 Performance criteria:

2.2.1 CPU Utilization We want to keep the CPU as busy as possible that means CPU is not free during the execution of processes. Conceptually the CPU utilization can range from 0 to 100 percent.

2.2.2 Throughput: If the CPU is executing processes, then work is being completed. One measure work is the number of processes that are completed per time unit that means the number of tasks per second which the scheduler manages to complete the tasks.

2.2.3 Response Time: In an interactive system, turnaround time may not be best measure. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, response time is the time from the submission of a request until the first response is produced that means when the task is submitted until the first response is received. So the response time should be low for best scheduling.

2.2.4 Turnaround Time: Turnaround time refers to the total time which is spend to complete the process and is how long it takes the time to execute that process. The time interval from the time of submission of a process to the time of completion is the turnaround time. Total turnaround time is calculation is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

2.2.5 Waiting Time: The waiting time is not the measurement of time when a process executes or does I/O completion; it affects only the amount of time of submission of a process spends waiting in the ready queue. So the Waiting time is the period of spent waiting in the ready queue to submit the new arriving process for the CPU.

2.2.6 Correctness: For algorithms which permit the user to attach deadlines to a task, this measures the proportion of time or proportion of total scheduling time unit when a task fails to meet its deadline that means all the task should be meet at the given deadline.

2.2.7 Overhead: Overhead refers to the proportion of time wasted due to computation of the schedule, and the system overhead due to context switching the tasks when new task is arriving.

2.2.8 Preventability: Predictability refers to the degree to which a task runs in a conventional manner that is it takes approximately the same time or with the same cost, irrespective of the load on the system.

2.2.9 Efficiency: Efficiency refers to the respective of system when CPU is busy for scheduling of new arriving tasks.

2.2.10 Fairness: In the absence of user or system supplied criteria for selection, the scheduler should allocate a fair amount of the resource to each task.

2.2.11 Load **balancing**: The scheduler should balance the load across other system resources, such as memory and I/O usage.

2.3 Scheduling Algorithm

The CPU scheduler executes the processes when they schedules on it. When there are number of processes in the ready queue, the algorithm which decides the order of execution of those processes is called *scheduling algorithm*. The various well known CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF) and Priority scheduling. All the above algorithms are non-preemptive in nature and are not suitable for time sharing systems. Shortest Remaining Time First (SRTF) and Round Robin (RR) are preemptive in nature. RR is most suitable for time sharing systems[1].

3. OUR PROPOSED ALGORITHM

The Adaptive 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.

3.1 Intelligent Time Slice for Adaptive 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[6]; 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.

3.2 Adaptive 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 ith 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
```

4. EXPERIMAENTAL ANALYSIS

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.

Case 1:

We Assume five processes arriving at time = 0, with increasing burst time (P1 = 14, P2 =34, P3 = 45, P4 = 62, P5= 77) with time quantum =25 as shown in Table3.1. The Table3.2 shows the output using RR algorithm and Adaptive RR algorithm.Figure3.1 and Figure3.2 shows Gantt chart for both the algorithms simple RR and Adaptive RR respectively.

Process	Arrival Time (ms)	Burst Time (ms)

P1	0	14
P2	0	34
P3	0	45
P4	0	62
P5	0	77

Table3.1

According to simple RR

P1	P2	P3	P4	P5	P2	P3	P4	P5	P4	P5	P5

0 14 39 64 89 114 123 143 168 193 205 230 232

Figure 3.1 Ghantt Chart for Simple RR

According to proposed mechanism:

First of all I arrange the processes in ready queue according their given burst time in increasing order that is P1=14, P2=34, P3=45, P4=62 and P5=77 and after that I choosing the time quantum according Adaptive RR algorithm, the time quantum is the mid process burst time if the given processes are odd, that is 45.The Gantt chart for Adaptive RR

P1	P2	P3	P4	P5	P4	P5

0 14 48 93 138 183 200 232

Figure 3.2 Gantt Chart for Adaptive RR

Algorithm	Time Quantum	CS	Average WT	Average TAT	Throughput
Simple RR	25	11	97	143.4	low
Adaptive RR	45	7	71	117.4	high

Table3.2 Comparison of simple RR and Adaptive RR

Case 2:

We Assume five processes arriving at time = 0, with decreasing burst time (P1 = 83, P2 =54, P3 = 30, P4 = 19, P5= 8) with time quantum = 26 as shown in Table3.3. The Table3.4 shows the output using RR algorithm and Adaptive RR algorithm.Figure3.3 and Figure3.4 shows Gantt chart for both the algorithms respectively.

Process	Arrival Time (ms)	Burst Time (ms)
P1	0	83
P2	0	54
P3	0	30
P4	0	19
P5	0	8

Table3.3

According to simple RR

P1	P2	P3	P4	P 5	P1	P2	P3	P1	P 2	P1

0 26 52 78 97 105 131 157 161 187 189 194

Figure 3.3 Ghantt Chart for simple RR

According to proposed mechanism:

First of all I arrange the processes in ready queue according their given burst time in increasing order that is P5=8, P4=19, P3=30, P2=54 and P5=83 and after that I choosing the time quantum according Adaptive RR algorithm, the time quantum is the mid process burst time if the given processes are odd, that is 30.The Ghantt chart for Adaptive RR

P5	P4	P3	P2	P1	P2	P1	P1

0 8 27 57 87 117 141 171 194

Figure 3.4 Ghantt Chart for Adaptive RR

Algorithm	Time Quantum	CS	Average WT	Average TAT	Throughput
Simple RR	26	11	110.4	149.2	low
Adaptive RR	45	7	46.6	85.4	high

Table3.4 Comparison of simple RR and Adaptive RR

5. CONCLUSION

This paper gives the better result comparison to Simple Round Robin Scheduling Algorithm and solves the problem of shortest job first scheduling algorithm which is starvation. This approach also solves the problem higher average waiting time of first come first serve scheduling algorithm. The result performance I had shown in the performance chart above. The performance in the reference of average waiting time, turnaround time and context switches. For the future perspective the research should be useful with the knowing of arrival time with burst time and you can analysis this research for improvement.

6. 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.