Sushruta Mishra et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 4, Issue 3, Sept 2017, pp. 221-225

CPU Scheduling using an Optimized Round-Robin Scheduling Technique

Sushruta Mishra[1], Soumya Sahoo[2], Sunil Mohapatra[3], Brojo Kishore Mishra[4],

[1,2,3]Dept. of CSE, C. V. Raman College of Engineering, Bhubaneswar, INDIA
[4]Dept. of IT, C. V. Raman College of Engineering, Bhubaneswar, INDIA

Abstract- **One of the most vital tasks of Operating System is to manage different processes in memory such that the CPU utilization is optimized. In order to maximize CPU utilization processes are required to be scheduled in an efficient manner so that maximum numbers of processes are provided the services of CPU. There exist several Scheduling schemes to schedule processes efficiently. Every scheme has its unique properties. In this paper we have proposed a new Scheduling technique which is based on Round-Robin Scheduling and have demonstrated and proved experimentally that our proposed scheme is more efficient than Round-Robin Scheduling by minimizing the total Turnaround time and the Average Waiting time.**

***Keywords*— CPU Scheduling, Round-Robin Scheduling, Average Waiting time, Gantt chart**

I. INTRODUCTION

Operating system is a system program which acts as an interface between the user and the computer. It provides an environment where user can execute their application programs. Thus Operating system is the core part in computer based architecture which performs several vital tasks like process management, memory management, file management, I/O management, protection and security etc. Among these functionalities process management is the most critical job of operating systems. Operating system performs the following tasks related to process management.

- *Creating and deleting user processes*
- *Suspending and resuming processes*
- *Process synchronization*
- *Inter Process communication*
- *Deadlock handling*

There are multiple processes in memory ready for execution but at a given instant one process can be allocated the CPU and this task is done by scheduling algorithms which permits the CPU to switch from one process to another and allocates them as per the requirement of the user. Thus process management means managing and controlling processes in an efficient manner so that system throughput is enhanced. Thus for efficient process management there is need to manage and coordinate CPU in a systematic manner because it is the CPU which is responsible to manage all processes. So to improve the efficiency of CPU so that it can control as many processes in memory various scheduling algorithms are used. There are many algorithms available for CPU scheduling and every algorithm has its own merits and demerits. Operating system should allow process many as possible running at all times in order to maximize the CPU utilization. In a multi-programmed operating System a process is executed until it must wait for the completion of some I/O request. In this case the time has been used proficiently. A number of processes are kept in memory simultaneously and while one process occupies the CPU selected by the scheduler. Thus CPU scheduling is central to Operating system design. CPU scheduling determines which process run when there are multiple runnable processes. CPU scheduling is important because it can have a big effect on resources utilization and overall performance of the system.

II. SCHEDULING CRITERIA

Several scheduling algorithms exist and every scheduling algorithm have its own distinctive properties thus in order to choose a particular scheduling algorithm various properties are to be considered. Many criteria have been suggested for comparing CPU scheduling algorithms which includes the following.

- *CPU utilization:* It refers that the CPU should be kept busy as long as possible. In less loaded systems it should be around 40% while in heavily loaded systems it should range upto 90% approximately.
- *Throughput:* When the CPU is busy executing various processes then substantial amount of work is done. Thus this amount of work done implies the number of processes that completes its execution. Throughput is denoted by the number of processes completed per unit time.
- *Turnaround time:* An important criteria is the duration of time that a process takes to complete. Thus the interval from the time of submission of a process to the time of completion is the turnaround time.
- *Waiting time:* The CPU can execute only one process at a time while others have to wait in the ready queue for their turn. Sum of the periods spent waiting in the ready queue is the waiting time.
- *Response time:* In an interactive system, turnaround time may not be the best criterion. 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, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the

time it takes to start responding, not the time it takes to output the response.

A good scheduling algorithm should optimize the above performance measures. The optimization performance measures are:

- Maximize CPU utilization
- Maximize throughput
- Minimize turnaround time
- Minimize waiting time
- Minimize response time
- Maximize scheduler efficiency

III. RELATED WORK

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many existing scheduling algorithms. First-Come First-Served Scheduling, Shortest-Job-First Scheduling, Round-Robin Scheduling, Priority based Scheduling are some of the general and popular scheduling algorithms used.

First-Come First-Served Scheduling: Here the process that requests the CPU first is allocated the CPU first.

Shortest-Job-First Scheduling: This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS algorithm is used to break the tie.

Round-Robin Scheduling: In this algorithm a time quantum is defined and the ready queue is treated as a circular queue. The CPU scheduler moves around the circular queue allocating the CPU to each process for a time interval of up to 1 time quantum.

Priority based Scheduling: Here a priority is attached to each process and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS order.

Our proposed algorithm is based on Round-Robin Scheduling algorithm. This algorithm is designed for time-sharing systems. It is similar to FCFS scheduling but preemption is added to switch between processes. It allocates the CPU to the first process in the ready queue for q time units, where q is the time quantum. After q time units, if the process has not relinquished the CPU, it is preempted and the process is put at the tail of the ready queue. If the quantum is too large this algorithm degenerates to FCFS scheduling. Round-Robin Scheduling algorithm can be better understood by the following steps:

1. A queue is maintained by the CPU scheduler that consists of a series of ready processes and a list of swapped out and blocked processes.
2. PCB of the newly created process is added to the end of the ready queue and the PCB of the terminated process is removed from the ready queue.
3. The scheduler always selects the PCB at the head of the ready queue.
4. After a running process completes its time slice it is moved to the end of the ready queue.
5. The event handler perform the following action a) When a process makes an I/O request or swapped out, its PCB is removed from the ready queue to blocked/swapped out list. b) When I/O operation awaited by a process finishes or process is swapped in its PCB is removed from blocked/swapped list to the end of the ready queue.

IV. PROPOSED WORK

Our proposed algorithm is based on Round-Robin Scheduling with an enhanced performance. The algorithm is described as follows.

1. *Allocate the CPU to all processes only once like in general Round-Robin scheduling algorithm.*
2. *After first round shortest process from the waiting queue is selected and is allocated to the CPU.*
3. *After that the next shortest job is selected and step 2 is repeated.*
4. *Steps 2 and 3 are repeated until all processes have completed execution*

V. EXPERIMENTAL RESULTS

Our proposed Round-Robin Scheduling is a combination of SJF (shortest job first) scheduling algorithm and Round-Robin scheduling algorithm. We have verified this by performing mathematical computations by implementing Gantt chart. Here we will discuss two experiments based on which results are obtained. Here I have two types of problems one is with arrival time another is without arrival time. Let the time slice be 2. The state of processes is given in the table below.

Process	A	B	C	D
Arrival time	0.000	1.001	4.001	6.001
Burst time	3	6	4	2

Now according to our general Round-Robin Scheduling algorithm:

Gantt chart =

A	B	A	B	C	D	B	C

Average turnaround time is =8.25 ms
Total Waiting Time=18 ms

Now according to our proposed algorithm:
Gantt chart =

A	B	A	B	B	D	C	C

Total waiting Time =14 ms
Average turnaround time=7.5 ms

Now if the arrival time is not considered and the time quantum is taken as 5 then:

Process	A	B	C	D	E
Burst time	24	20	8	10	3

Now according to our general Round-Robin Scheduling algorithm:
Gantt chart =

A	B	C	D	E	A	B	C	D	A	B	A

Average turnaround time is =8.25 ms
Total Waiting Time=18 ms

Now according to our proposed algorithm:
Gantt chart =

A	B	C	D	E	C	D	B	B	B	A	A	A	A

Total Waiting time=131 ms
Average turnaround time= 39.2 ms

Thus we can see from the above Gantt chart that total waiting time and average turnaround time both are reduced by using our proposed algorithm. The reduction of total waiting time and turnaround time shows maximum CPU utilization that and minimum response time. Hence it is verified that our proposed algorithm proves to be much more efficient compared to the general Round-Robin scheduling algorithm.

VI. CONCLUSION

Thus as we demonstrated that our proposed scheme is a time-efficient algorithm and can be implemented as a CPU scheduling algorithm effectively. Thus in future the efficiency of time sharing systems can be significantly improved by using this algorithm.

VII. REFERENCES

[1]. Sabrina, F.C.D, Nguyen, S.Jha, D. Platt and F. Safaei, 2005. Processing resources scheduling in programmable networks. Computer commune, 28:676-687.
[2]. Silberchatz, Galvin and Gagne, 2003 .operating systemsconcepts,(6th edn, John Wiley and Sons)
[3]. Seltzer, M P. Chen and J outerhout, 1990.Disk scheduling revisited in USENIX. Winter technical conference.
[4]. Shamim H M 1998. Operating system, DCSA 2302. School of sciences and Technology. Bangladesh Open University Gazipur-1705
[5]. D. M. Dhamdhere Operating Systems A Concept Based Approach**,** Second edition**,** Tata McGraw-Hill, 2006
[6]. Operating Systems Sibsankar Haldar 2009, Pearson Education, India

