International Journal of Latest Trends in Engineering and Technology
Special Issue SACAIM 2017, pp. 510-513
e-ISSN:2278-621X

COMPARISON OF SCHEDULING ALGORITHMS WITH REFERENCE TO THE NUMBER OF CONTEXT SWITCHES IN MULTITHREADING

Jovel Cardoso[1], Nelton Goes[2] & Mervin Rodrigues[3]

Abstract- Multithreading is important but the cost of context switching is high. The reason for context switching is pre-emption which causes large amount of memory usage in the system. This paper focuses on comparing the scheduling algorithms to find out which gives maximum system throughput and responsiveness. The algorithms that are used to study are round robin priority based algorithm, Shortest remaining time Scheduling.
Keywords -Context Switches, Pre-emption, System responsiveness

1. INTRODUCTION

In Multithreading, the processes are divided into pieces which are known as threads and the "kernel" helps them to run. Multithreaded program has two or more parts that run side by side. Since multiple threads run simultaneously in multithreading it can be said that as multitasking. Multitasking can be categorized into two that is process-based and thread-based multitasking. In the process-based multitasking two or more processes are executed by the processor simultaneously.
In thread -based multitasking, a single unit of code is considered as a task, which means a single program can perform more than one task. Overheads of multitasking processes are more than multitasking threads. Processes generally require their own address space and are considered as heavyweight. So, the context switching from one process to another is costlier. Communication between the processes is limited and expensive. Threads share the same address space and are lightweight. In thread based, the communication between threads is inexpensive, and context switching from one thread to the next is cost efficient. The paper mainly focuses on comparing the preemptive scheduling algorithms based on context switching. The advantages of preemptive scheduling over non-preemptive scheduling in particular are flexibility, improved processor utilization etc. but it results in extra CPU cycles and memory requirements due to unnecessary context switches. If the context switches are not regulated, they may result in reduced responsiveness, unnecessary delays, wastage of energy and extra memory requirements. All these may produce high overhead in real time embedded systems. In this paper, two algorithms are compared with reference to context switching in multithreading. The first algorithm used is the Round Robin scheduling and second algorithm used is shortest remaining time first to allocate the processor to various threads, the average of the burst time of various threads are calculated and the average value is kept as a new time quantum which is ultimately assigned to each thread.

2. RELATED WORK

In recent years, many new algorithms have been developed and the existing algorithms are modified so that they the performance increases. A priority based round robin algorithm allocates CPU to every process in round robin fashion, according to the given priority, for given time quantum. After the first step, it arranges it increasing order or their remaining CPU burst time in the read queue. New priorities are assigned according to the remaining CPU burst of processes. Reducing the number of context switching through a proposed algorithm divide and conquer where all the processes are divided into check points and round robin algorithm is applied to that. Comparative study can be of various algorithms based on CPU utilization, waiting time, turnaround time, response time and throughput. Optimized Short Job First Scheduling Algorithm it is combination of two algorithms that is Shortest Job First and the other one is the constrained on remaining burst time of running process. It is a preemption algorithm but it decides before preemption. If half of the remaining executing time of running process is less than the burst time of newly process, then running process will not preempt. If half of the remaining execution time of therunning process is greater than the burst time of the newly created process, then we will preempt running process and assign processor to newly arrived process. Improved Mean Round Robin with Shortest Job First Scheduling which improves the efficiency and performance of round robin degrades with respect to context switching.

[1] MCA III Semester,AIMIT,St Aloysius College(Autonomous), Mangalore, Karnataka, India
[2] MCA III Semester,AIMIT, St Aloysius College(Autonomous), Mangalore, Karnataka, India
[3] MCA III Semester,AIMIT, St Aloysius College(Autonomous), Mangalore, Karnataka, India

3. ROUND ROBIN SCHEDULING ALGORITHM

Round-robin (RR) is one of the algorithms employed by Process Scheduler and Network Scheduler in Computing. As the term interprets, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, where the processes are handled without priority (also known as cyclic executive).Round-robin scheduling is simple, easy to implement and starvation-free. Round-robin scheduling can also be applied to other scheduling issues, such in computer networks for data packet scheduling. To give each job a slot or quantum, a round-robin scheduler employs time-sharing, and the job is interrupted if it is not completed by them. The next time a time slot is assigned to that process, the job is resumed. If the process changes or terminates its state for waiting during its quantum period of time, the scheduler selects the first process from the ready queue to execute. The process that produced large jobs would be favored over other processes, in the absence of time-sharing or if the quantum time were related to the size of the job. Since the scheduler forces the process out of the CPU once the time expires it is a preemptive algorithm.

4.SHORTEST REMAINING TIME FIRST ALGORITHM

Here the Process with the smallest amount of time remaining for completion of its execution is selected to execute. Since the process which is currentlyexecuting has theshortest amount of time remaining and that time should only reduce when execution continues, processes will continue to run until they complete or a new process is added that requires a smaller amount of time than the ones already present.

The shorter the time remaining, it is advantageous because short processes are handled very quickly. Since the system makes a decision when a new process is added or a process completes and so it requires very little overhead. When a new process is added to the algorithm, it only needs to compare the currently executing process with the newly added process and ignore all other processes which are waiting for its execution.

5.COMPARISON CONDITIONS

5.1 Burst time :

During scheduling, each process is granted access to use the CPU for its time slice. The time slice that the process gets is called the CPU burst. In simple terms, the total time a process gets control of the CPU is the CPU burst time, and this concept of gaining control of the CPU is called the CPU burst.

5.2 Waiting time :

CPU scheduling is a process which allows only one process to use the CPU at a time while the other processes are put to the waiting state due to non-availability of any required resource, therefor making full use of the CPU. The main aim of CPU scheduling is to make the system efficient, fast and fair.

5.3 Turnaround Time :

The total time taken between the submission of a program or process or thread or task for execution and the return of the complete output to the customer or user is called the Turnaround Time. It may change for different programming languages depending on the developer of the software or the program. Turnaround time simply deals with the total time taken by the program to provide the required output to the user after the program is started.

Process Name	Arrival time	Burst time
T0	0	250
T1	50	170
T2	130	75
T3	190	100
T4	210	130
T5	350	50

Table -1 Arrival and Burst time of the threads.

Shortest time remaining first algorithm

Process	Waiting time(ms)	Turnaround time(ms)
1	525	775
2	75	245
3	0	75
4	105	205
5	235	365
6	45	95

Table -2 Waiting time and turnaround time using shortest time remaining first algorithm

Process of execution

T0	T1	T2	T1	T3	T5	T4	T0

Number of context switches:7
Average waiting time:164.16667ms
Average Turnaround time:293.33334ms

Round Robin algorithm

Process	Waiting time(ms)	Turnaround time(ms)
1	525	775
2	475	645
3	70	145
4	85	185
5	385	515
6	125	175

Table -3 Waiting time and turnaround time using round robin with Quantum time 100ms

Process of execution

T0	T1	T2	T3	T4	T5	T0	T1	T4	T0

Number of context switches: 9
Average waiting time: 277.500ms
Average Turnaround time: 406.6666ms

Process	Waiting time(ms)	Turnaround time(ms)
1	525	775
2	150	320
3	240	315
4	255	355
5	335	465
6	325	375

Table -4 Waiting time and turnaround time using round robin with Quantum time 200ms

Process of execution

T0	T1	T2	T3	T4	T5	T0

Number of context switches: 6
Average waiting time: 305.0000ms
Average Turnaround time: 434.166667ms

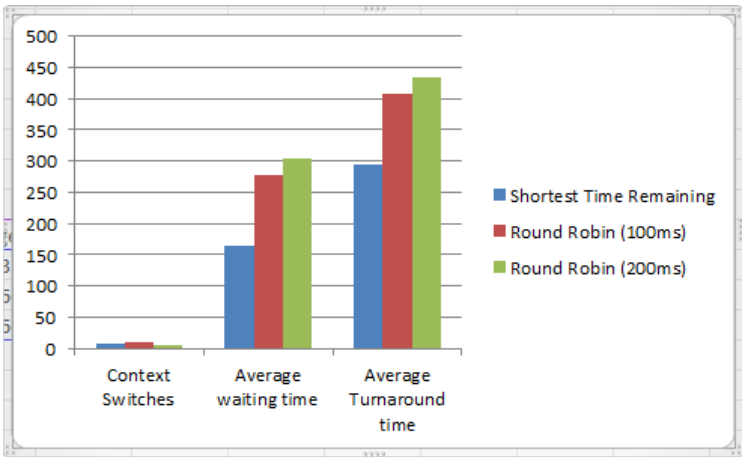

Fig-1 shows context switches, average waiting time and average turnaround time

6. CONCLUSION

The number of context switching in round robin is more when compared to shortest remaining scheduling algorithm but some process in shortest time remaining algorithm have to wait for a long period to run which can be considered as starvation. Number of context switching increases in shortest time remaining algorithm if more processes with less burst time arrive. In round robin algorithm if the quantum time is increased the number of context switching decreases.

7. REFERENCES

[1] Ishwari Singh Rajput, Deepa Gupta "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems,".Available:https://www.idconline.com/technical_references/pdfs/information_technology/A%20Priority%20based.pdf

[2] Tarun Sharma Tapsi Sharma Sweta Bhattacharya, "Reducing the Number of Context Switches in Multithreading," International Journal of Advanced Research in Computer Science and Software Engineering.

[3] "http://zeus.cs.pacificu.edu/ryand/cs460/2014/Lectures/ch6bModified.pdf"

[4] LalitKishor, Dinesh Goyal, "Comparative Analysis of Various Scheduling Algorithms Available: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-2-ISSUE-4-1488-1491.pdf

[5] Muhammad Akhtar1 , Bushra Hamid1 , Inayat ur-Rehman2 , Mamoona Humayun1 , Maryam Hamayun1 and Hira Khurshid1 "An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling".Journal of Applied Environmental and Biological Sciences. Available: https://www.textroad.com/pdf/JAEBS/J.%20Appl.%20Environ.%20Biol.%20Sci.,%205(12)42-46,%202015.pdf"

[6] RadheShyam, Sunil Kumar Nandal, "Improved Mean Round Robin with Shortest Job First Scheduling" International Journal of Advanced Research in Computer Science and Software Engineering. Available: https://www.ijarcsse.com/docs/papers/Volume_4/7_July2014/V4I7-0103.pdf