International Journal of Computer Applications (0975 – 8887)
Volume 12– No.3, November 2010

Simulator for Performance Evaluation of Process Scheduling Policies for Embedded Real-Time Operating Systems

Rajesh Kumar Garg
Lecturer,Electronic & Communication Engg.
Seth Jai Parkash Polytechnic
Damla, Yamuna Nagar-135001
Haryana (INDIA)

Vikram Singh
Professor
Deptt of Computer Science & Applications
Chaudhary Devi Lal University, Sirsa
Haryana (INDIA)

ABSTRACT: The embedded systems work on real time operating systems (ERTOS). There are many CPU scheduling policies in general operating systems out of which ERTOS most commonly follow the priority and round robin scheduling. The aim of this research work is to evaluate performance of these scheduling policies. Simulation is adopted as tool to find the best policy that can be implemented to boost the performance of the ERTOS. This model is in the form of a set of assumptions concerning operation of a system. The simulator designed accesses the performances of Round Robin, Priority preemptive and non preemptive scheduling policies in terms of average waiting time and average turnaround time for a number of processes.

Key words: - Simulator, Scheduling, Round Robin, Priority Scheduling, Embedded Real Operating system

I INTRODUCTION

Scheduling is the activity of selecting the next request to be handled by a server. In the classical view of scheduling in operating systems, the server is the CPU of the system and requests represents executions of user computations. The quality of service provided to requests and utilization of resources in the system is determined by the manner and the order in which the requests are serviced. This fact gives rise to two fundamental techniques used in servers. (Tannenbaum et. al, 2003) 1. Preemption of requests- A request being processed is preempted in order to provide fair service to all the requests and 2.Reordering of requests:- Requests are reordered to improve the average quality of service provided by a server. In some environments servicing shorter requests first increases the rate at which a server completes requests. Scheduling decision for embedded software play an important role on system performance. The designer should select the right scheduling algorithm at high abstraction levels so as to save him from the error-prone and time consuming task of tuning code delays or task priority assignments at the final stage of system design The main objective of this research is to find out the best Scheduling policy for ERTOS (Ramamritham & Stankovic, 1994)

CPU SCHEDULER Whenever the CPU becomes idle; the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in the memory that are ready to execute, and allocates the CPU to one of them Fig.1 shows a schematic of scheduling.

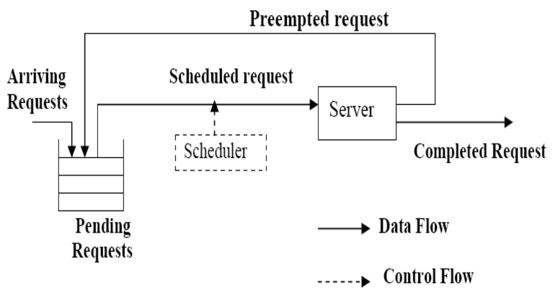

Fig 1.

The ready queue is not necessarily a first-in, first-out (FIFO) queue. It may be implemented as a FIFO queue, a priority queue, a tree or simply an unordered linked list. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. (Stallings William, 2004)An operating system must allocate computer resources among the potentially competing requirements of multiple processes. In the case of the processor, the resource to be allocated is execution time on the processor and the means of allocation is scheduling. This way, the scheduler is the component of the operating system responsible to grant the right to CPU access to a list of several processes ready to execute. This idea is illustrated in the five-state diagram of Figure 2. (Galvin et. al, 2001)

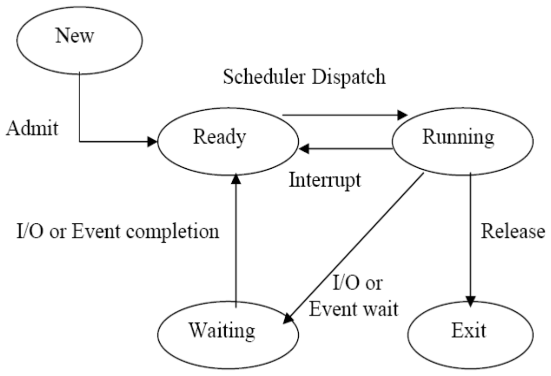

Fig 2

In circumstances first and fourth, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution. There is a choice, however, in circumstances second and third. When scheduling takes place under circumstances first and fourth, then the scheduling scheme is non-preemptive; otherwise, the scheduling scheme is preemptive.

Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.

II. TYPES OF SCHEDULING

We have designed simulator for two types of CPU scheduling. One is round robin (RR) and other is priority type.

Round-Robin Scheduling The CPU scheduler goes around the ready queue allocating the CPU to each process for a time interval of up to one time slice or time quantum. A small unit of time, called a time quantum (or time slice), is defined. A time quantum is generally from 10 to 100 milliseconds. The ready queue, which is treated as a circular queue, is kept as First in First out (FIFO) queue of processes. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum. One of the two things will happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. (Yaashuwanth et.al, 2010)] New processes are added at the tail of the ready queue and CPU scheduler picks the process from the head of the queue. After expiry of the time quantum the process is preempted and placed at the tail of the ready queue. In real systems, however, the Context switching would take place on expiry of each time slice. RR scheduling achieves equitable sharing of system resources. Short processes may be executed within a single time quantum and thus exhibit good response time. Long processes may require several quanta and thus be forced to cycle through the ready queue for a longer time before completion Otherwise, if the CPU burst of the currently running process is longer than 1 time quantum; the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

Priority Scheduling There are two types of priority scheduling. It can be preemptive or Non-preemptive.

Preemptive Priority Scheduling In this type of scheduling the CPU is allocated to the process with the highest priority immediately upon the arrival of the highest priority process. If the equal priority process is in running state, after the completion of the present running process CPU is allocated to this even though one more equal priority process is to arrive. When a process arrives at the ready queue, its priority is compared with the priority of the currently running processes. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process. (Deital,1984)] and (Dhamdhere,2003)**.** Priorities are generally some fixed range of numbers and low numbers represents high priority. (Liu and Layland, 1973) were perhaps the first to formally study priority-driven algorithms.

Non-preemptive Priority Scheduling In this type of scheduling the CPU is allocated to the process with the highest priority after completing the present running process. A non preemptive priority-scheduling algorithm will simply put the new process at the head of the ready queue

III. DESIGN OF SIMULATOR

RTOS Scheduling Algorithm Simulator (RSAS) has been developed using C++ language under windows operating system on an Intel compatible machine with a view to develop a software tool, which can be used for the study and evaluation of CPU scheduling algorithms in real time operating systems. The system discussed here is stochastic and dynamic in nature. (Robinsatein et.al.,1998). The next –event discrete simulation model (Averill M, 2008). has been used for conducting simulation experiment. RSAS has been developed as a comprehensive software package which runs a simulation in real time, generates useful data to be used for evaluation and provides a user-friendly environment. It reads simulation data generated randomly into a linear array. In accordance with the data it makes Process Control Blocks (PCB) and develops ready queue in memory. On user instruction, the control is then handed over to relevant simulation module RR gives intermediate wait and turnaround time. In software, it is needed to consider the effect of context switching on the performance of RR scheduling. Thus, it is wanted that the time quantum to be large with respect to the context-switch time. Turn around time also depends on the size of the time quantum. The burst time is generated using exponential distribution with mean arrival rate of 5. The processes arrival time is also generated using same distribution but with inter arrival rate 6.5.The priority of the process is generated using formula P(i)=4*(i+1)/6 and substituting this value into exponential distribution in place of mean arrival .

IV RESULTS AND DISCUSSIONS

Table 1 shows the no. of processes; CPU burst time, arrival time and priority of 20 processes. Table 5 shows the no. of processes; CPU burst time, and arrival time and priority of 30 processes. The time quantum for round robin is taken as 7. Tables show the output of Simulator for different types of scheduling techniques. First 20 and second 30 processes are considered and out put are taken. The parameter like average waiting time and average turnaround time is measured (see Tables 2, 3, 4) and for 30 (see Tables 5, 6, 7, 8) The average waiting time for preemptive scheduling is 38.1 ms and average turnaround time is 45 ms. For 30 processes these values are 40.13 ms and 46.16 ms. For 20 processes the average waiting time for non preemptive scheduling is 38.1 ms and average turnaround time is 45 ms. For 30 processes these values are 39.66 ms and 45.7 ms.

The average waiting time for 20 processes round robin scheduling is 77.4 ms and average turnaround time is 53.4 ms. For 30 processes these values are 94.6333 ms and 49.9 ms.

Total No. Processes=20
Table 1 Different parameters of
Priority Scheduling

Process Information			
Process	Burst Time	Arrival Time	Priority
1	14	0	1
2	2	7	2
3	2	10	1
4	7	11	1
5	7	20	3
6	8	20	1
7	14	26	3
8	1	31	8
9	19	32	6
10	3	32	4
11	7	32	20
12	7	33	4
13	7	37	4
14	5	40	12
15	8	41	7
16	2	41	12
17	11	45	4
18	1	50	4
19	5	55	22
20	8	55	62

Table 2 Different parameters of Preemptive Priority Scheduling

Preemptive Priority Scheduling		
Process	Waiting Time	Turnaround time
1	0	14
2	24	26
3	4	6
4	5	12
5	13	20
6	3	11
7	14	28
8	79	80
9	51	70
10	22	25
11	86	93
12	24	31
13	27	34
14	71	76
15	61	69
16	75	77
17	26	37
18	32	33
19	70	75
20	75	83
Average	38.1	45

Table 3 Different parameters of Non Preemptive Priority Scheduling

Non Preemptive Priority Scheduling				
Process	Start Time	Finish Time	Waiting Time	Turnaround time
1	0	14	0	14
2	31	33	24	26
3	14	16	4	6
4	16	23	5	12
5	33	40	13	20
6	23	31	3	11
7	40	54	14	28
8	110	111	79	80
9	83	102	51	70
10	54	57	22	25
11	118	125	86	93
12	57	64	24	31
13	64	71	27	34
14	111	116	71	76
15	102	110	61	69
16	116	118	75	77
17	71	82	26	37
18	82	83	32	33
19	125	130	70	75
20	130	138	75	83
		Average	38.1	45

Table 4
Different parameters of Round Robin Scheduling

Round Robin			
Process	Burst time	Waiting Time	Turnaround Time
1	14	107	121
2	2	9	4
3	2	11	3
4	7	13	9
5	7	20	7
6	8	27	15
7	14	142	130
8	1	44	14
9	19	164	151
10	3	54	25
11	7	57	32
12	7	64	38
13	7	71	41
14	5	78	43
15	8	83	50
16	2	91	52
17	11	200	166
18	1	102	53
19	5	103	53
20	8	108	61
	Average	77.4	53.4

Total No. Processes=30

Table 5 Different parameters of Priority Scheduling

Process information			
Process	Burst Time	Arrival Time	Priority
1	13	0	1
2	4	6	4
3	1	6	2
4	3	6	1
5	9	6	8
6	2	12	2
7	1	18	2
8	5	18	1
9	10	25	11
10	2	25	4
11	8	26	1
12	3	31	2
13	2	40	25
14	8	40	10
15	3	47	8
16	4	47	20
17	14	53	2
18	5	53	12
19	20	58	14
20	2	67	22
21	4	69	14
22	1	69	50
23	1	86	17
24	1	93	18
25	2	95	5
26	8	99	20
27	12	101	1
28	3	104	47
29	28	111	5
30	2	112	29

Table 6
Different parameters of Preemptive Priority Scheduling

Preemptive Priority Scheduling		
Process	Waiting Time	Turnaround time
1	0	13
2	30	34
3	10	11
4	7	10
5	36	45
6	10	12
7	6	7
8	0	5
9	51	61
10	15	17
11	0	8
12	3	6
13	133	135
14	28	36
15	18	21
16	112	116
17	0	14
18	33	38
19	75	95
20	104	106
21	84	88
22	111	112
23	71	72
24	65	66
25	0	2
26	64	72
27	0	12
28	73	76
29	2	30
30	63	65
Average	40.1333	46.1667

Table 7

Different parameters of Non Preemptive Priority Scheduling

Non Preemptive Priority Scheduling				
Process	Start Time	Finish Time	Waiting Time	Turnaround time
1	0	13	0	13
2	25	29	19	23
3	16	17	10	11
4	13	16	7	10
5	42	51	36	45
6	17	19	5	7
7	24	25	6	7
8	19	24	1	6
9	76	86	51	61
10	40	42	15	17
11	29	37	3	11
12	37	40	6	9
13	173	175	133	135
14	68	76	28	36
15	51	54	4	7
16	159	163	112	116
17	54	68	1	15
18	86	91	33	38
19	91	111	33	53
20	171	173	104	106
21	153	157	84	88
22	180	181	111	112
23	157	158	71	72
24	158	159	65	66
25	123	125	28	30
26	163	171	64	72
27	111	123	10	22
28	177	180	73	76
29	125	153	14	42
30	175	177	63	65
		Average	39.6667	45.7

Table 8

Different parameters of Round Robin Scheduling

Round Robin			
Process	Burst time	Waiting Time	Turnaround Time
1	13	129	142
2	4	9	7
3	1	13	8
4	3	14	11
5	9	17	20
6	2	26	16
7	1	28	11
8	5	29	16
9	10	163	148
10	2	43	20
11	8	45	27
12	3	53	25
13	2	56	18
14	8	58	26
15	3	66	22
16	4	69	26
17	14	202	163
18	5	82	34
19	20	238	200
20	2	96	31
21	4	98	33
22	1	102	34
23	1	103	18
24	1	104	12
25	2	105	12
26	8	107	16
27	12	244	155
28	3	124	23
29	28	280	197
30	2	136	26
	Average	94.6333	49.9

Table 9 Different Waiting Times different of different Scheduling

WAITING TIME			
Total Processes	Preemptive Priority Scheduling	Non Preemptive Priority Scheduling	Round Robin Scheduling
10	37	35.9	71.8
20	38.1	38.1	77.4
30	40.13	39	94.63

The Table 9 and Table 10 show the waiting time and turnaround time for total no of 10, 20 and 30 processes.

The output for three different scheduling processes has been shown. The round robin takes max average time and turnaround time. Figure 4 shows the bar diagram for different values of processes and their respective values for average waiting times.

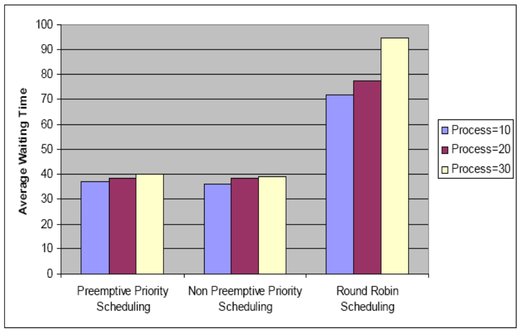

Figure 4 Average Waiting Times of different Scheduling Process

Table 10 Different Turnaround Times different of different Scheduling

TURNAROUND TIME			
Total Processes	Preemptive Priority Scheduling	Non Preemptive Priority Scheduling	Round Robin Scheduling
10	47.1	46	62.3
20	45	45	53.4
30	46.16	45.7	49.9

Table 10 shows the value of turnaround time for different no of processes. The turnaround time is max for round robin scheduling and approximately equal for non preemptive and preemptive. Fig 5 shows the bar diagram for average turnaround time for 10, 20 and 30 processes.

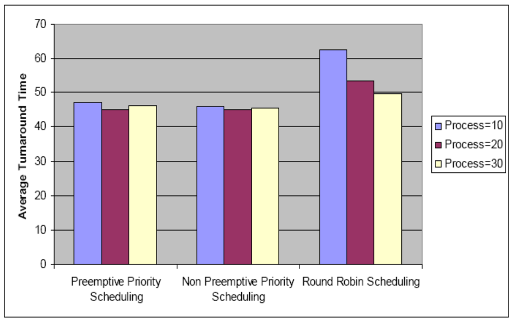

Figure 5 Average Turnaround Times of different Scheduling Process

V. CONCLUSION

Simulator (RSAS) has been developed. Ready queue is maintained as a FIFO queue in RR scheduling module. Processes are selected from the head of the ready queue. Processes being executed are preempted on expiry of time quantum, which is a user- defined data. A preempted process is linked at the tail of the ready queue. Process completing CPU burst before the expiry of time quanta either terminates. Priority scheduling module compares the priorities of processes where the priorities are the user-defined data. The process having highest priority is given the processor first and then the next level priority and so on up to the completion of the workload. Preemptive Scheduling response for the highest priority process is very good. But starvation may be possible for the lowest priority processes. The average waiting time under the Round Robin policy is often quite long.

REFERENCES

[1]Ramamritham, K.Stankovic,J.A. ,"Scheduling algorithms and operating systems support for real-time systems", January 1994

[2] Tannenbaum Andrew S. and Woodhull Albert S., "Operating Systems: Design and Implementation", 2nd Edition, Prentice Hall, India, 2003

[3]Stallings William, “Operating Systems: Internals and Design Principles”, 4th Edition, Prentice Hall, India, 2004

[4] Yaashuwanth .C, Dr. R. Ramesh, “Design of Real Time scheduler simulator and Development of Modified Round Robin architecture”, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.3, March 2010

[5]Deital H. M. 1984, “An Introduction to Operating Systems”, Rev. 1st ed. Reading, MA: Addition-Wesley

[6] C. L. Liu and J. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” J. Amer. Compt. Mach., vol. 20, no. 1, pp. 4M1, 1973.

[7]Silberschatz, A., P. B. Galvin et. al., “Operating System Concepts”, 6th Edition, 2001

[8]Dhamdhere D.M., “Operating Systems: Concept based Approach”, 2nd Edition, Tata
McGraw Hill, India, 2003

[9] Law, Averill M., Simulation modeling and Analysis, 4th ed., McGraw-Hill, New York, 2008