Real Time Computing Systems

Niharika Anand Sharma, Manu Bansal

ABSTRACT: The real time computing systems respond to input immediately therefore there are strict timing constraints that have to be met to get the correct output. Real time applications are expected to generate output in response to stimuli within some upper bound. Real time systems change its state in real time even after the controlling processor has stopped its execution. The real time application respond to the stimuli within a particular deadline. Scheduling is deciding how use the processor's time on the computer and to provide efficient service to all users it is an arrangement of performing functions at specified time. The intervals between each function have been defined by the algorithm to avoid any overlapping. The scheduling techniques has been used in order to achieve optimized results in real time. In the paper we analyze various scheduling techniques and observes the various issues on which there is still a need to work.

Key Words: Scheduling, resources, power, real time operations

1 INTRODUCTION:

SCHEDULING, in computing, means how the processes can be assigned on the available CPU(s). It is a key concept of multitasking, multiprocessing and real-time operating system design [1,3-4]. It is done by a means of scheduler and dispatcher. A *scheduler* is a person or machine that organizes or maintains schedules. In computing, scheduler is software program that arranges jobs or computer's operations in an appropriate order. A *dispatcher* is a module which gives control of CPU to the process selected by the scheduler. It is a decision making process that deals with the allocation of common resources to various tasks at different time periods to achieve multiple objectives. The resources and tasks can be of different forms in homogeneous/heterogeneous organization. Priorities have been associated with the tasks; each task has its due date and earliest dead line. Similarly objectives can also be in the different forms, they can be minimization of completion time of last task or minimization of number of tasks completed after their respective due dates [2,8-9]. In an industry it may mean to assign appropriate workers to do some task each day. It enables us to perform tasks which are done in routine automatically and efficiently.

Niharika Anand Sharma: Sr.Lecturer,Department of Electronics and Communication Engineering,MIET,Meerut,E-mail: niharikaanand84@gmail.com

Manu Bansal:Asst.Professor, Department of Electronics and Communication Engineering,Thapar University,Patiala

LITERATURE SURVEY:

The following sections show the work done by the various researchers in the field of scheduling for real time processors.

In 2003 Sanjoy K. Baruah et. al. [18] presented a schedulabiity analysis of Rate Monotonic (RM) scheduling algorithm to determine schedulabiity of periodic task set on a particular uniform multiprocessor. They considered periodic model of hard real time tasks with two parameters namely execution parameter (Ci) and period (Ti). They have provided the conditions of feasibility of the task system on a uniform multiprocessor platform. Also they proved that all the deadlines will be met if task set is scheduled using Rate Monotonic (RM) algorithm.

In 2008 Euiseong Seo et. al. [16] presented an energy efficient technique for scheduling real time tasks on multicore processors to lower the power consumption and increasing the throughput. They presented two techniques which modify existing techniques of unicore processors for multicore processors. The two techniques suggested by them are: (i) Dynamic Repartitioning algorithm, which dynamically balances the task loads multiple cores to minimize the power consumption during execution. (ii) Dynamic core scaling algorithm, which reduces leakage power consumption by adjusting the number of active cores. The simulation results show that 25% of energy consumed can be conserved by dynamic repartitioning and 40 % is conserved by dynamic core scaling.

In 2008 Ming Xiong et. al. [17] suggested deferrable scheduling algorithm for maintaining real time data freshness for minimizing the update workload by maintaining the temporal validity of real time data. The update transaction follows aperiodic task model and sampling times of update transaction jobs were deferred as late as possible.

In 2009, Fengxiang Zhang et. al. [15] presented schedulability analysis for real time systems with EDF scheduling which reduces the number of calculations of the processor demand of every task set and thus reduces the calculation times for the schedulable as well as unschedulable task sets. Experimental results show that 96% of task sets complete each schedulability test in less than 30 calculations of processor demand function (h(t)).

In 2010 Wann-Yun Shieh et. al. [12] suggested energy efficient tasks scheduling algorithm for the dual core real time systems to satisfy the low power consumption demands. Their main aim was to minimize the power consumption at the same time they maintain the same performance level of the dual core processor. They include two online and offline attacks and proposed an Integer linear programming approach for the optimal scheduling. The experimental results shown in their work was evident that the energy consumption was could be about 38% effectively by using heuristic algorithm.

In 2010 Marko Bertogna et. al. [13] suggested limited pre-emption earliest deadline first scheduling of sporadic task systems. Their algorithm is based on the amount of pre-emption. With no pre-emption there will be significant scheduling overhead and as each pre-emption leads to context switching, it causes increase in runtime overhead. In their paper, they suggested a limited pre-emption technique which can schedule all the systems that can be scheduled by fully preemptive algorithm but with limited runtime overhead. It is provided with a non pre-emption function, Q, which is monotonically non-increasing. It takes time to the deadline of executing job as input and gives us the time for which such job could be executed non pre-emptively. They performed large number of simulations by varying number of tasks, n, total utilization, U, and task parameters. For n=10 the average number of pre-emptions has been reduced by 20% as compared to normal EDF.

In 2010 Xian-Bo He et. al. [14] suggested an improved version of earlier deadline first (EDF) scheduling algorithm which was based on fuzzy inference system (FIS). Their algorithm was suitable for embedded real time systems in uncertain environments in order to minimize the deadline miss ratio. All the tasks were periodic and fuzzy set describes the task's criticality and deadline distance. The tasks which have higher criticality and shorter deadline distance have the higher priority and are scheduled first. The experimental results shown in their paper were evident that the total deadline miss ratio was reduced compared to traditional EDF. Important tasks in improved EDF never miss their deadlines when the system is not overloaded.

In 2011 A. Burns et. al. [11] suggested partitioned EDF scheduling for multiprocessors using a new task splitting scheme. They have compared C=D splitting scheme with fully partitioned scheme. They suggested that for m processors there can be splitting of at most m-1 tasks. No special run time mechanisms were required and overheads were kept minimal. In fully partitioned systems, there is a problem of bin-packing due to overloading of processors. The suggested scheme was straightforward to implement without any unusual real time operating system functions. They considered that each processor used standard EDF policy and tasks have dynamic priorities. Evaluation of the technique showed that there was a remarkable increase in the performance. The average utilization factor of full processors with 8 tasks and with a total utilization of 4 had increased from less than 0.89 to 0.99.

3 MOTIVATION:

Real time systems are the computing systems that must react within the precise time constraints to events in the environment. The efficient scheduling of tasks on these systems becomes necessary in order to receive accurate and timely response. From the work done by various researchers it has been observed that there is still a need to optimize scheduling techniques so that the various constraints of real time systems can be met and performance can be enhanced.

4 Objective of this work:

1. To study various algorithms used for scheduling processes on a processor.
2. To study various techniques used for scheduling real time tasks.

3. Comparison of different Real Time scheduling algorithms.

4.1 Objectives of Scheduling:

There are some goals that must be achieved in order to perfectly schedule the task on the processor. Some of those objectives are mentioned below.

4.1.1 Fairness:

It is important goal to achieve under all circumstances. A scheduler guarantees that each process should get a fair amount of CPU time for its execution and there should be no condition of starvation. But giving equal or equivalent time to all the processes is also not fair as different processes are not equally critical.

4.1.2 Throughput:

It is the amount of work a computer can do in a period of time. The scheduler aims to maximize the number of jobs completed or processed per unit time.

4.1.3 Turnaround Time:

It is the amount of time taken for the process to get executed. It is the interval from the time of submission of a process to the time of completion of the process.
Turnaround time, Tr = Ts + Tw
Ts = Execution Time, Tw = Waiting time
The scheduler should minimize the turnaround time, but it is not upto the scheduler to affect the execution time but it can minimize the waiting time so as to reduce the turnaround time [3,8].

4.1.4 Waiting Time:

Similar processes are allocated equal CPU time or times are divided according to processes' priority. The time every process waits for its execution in the ready queue is known as waiting time and it is the duty of scheduler to minimize this time [3].

4.1.5 Efficiency

It signifies the amount of time the system is performing tasks. Scheduler should keep the system busy for most of the time when possible. If both CPU and I/O devices are working all the time then more work gets done per unit time [5, 9]. Usually the above goals conflict with each other, as amount of the CPU time is finite, so there is a need of trade-off between the achievable goals. Every scheduling technique aim to achieve either of above mentioned goals. For scheduling tasks on a processing unit, there can be an optimization for power along with clock period. There are various techniques devised to optimize power and processing speed.

5 TYPES OF OPERATING SYSTEM SCHEDULING

Basic operating systems schedulers can be of three distinct types. Scheduling types are divided according to the schedulers and the names of schedulers suggest the relative frequency with which these functions are performed [3,6]. The schedulers are:

5.1 Long-term or high-level scheduling:

It decides which processes are to be added to the set currently executing processes and which are to be exited. It controls the degree of multiprogramming in multitasking systems with a need of trade off between degree of multiprogramming and throughput. The higher the number of processes, the smaller the time each of the processes may control CPU for. Long-term scheduler is also known as Admission Scheduler [7]. It is required to ensure that real time processes get enough CPU time to complete their tasks. The long term queue exists in the hard disk or virtual memory.

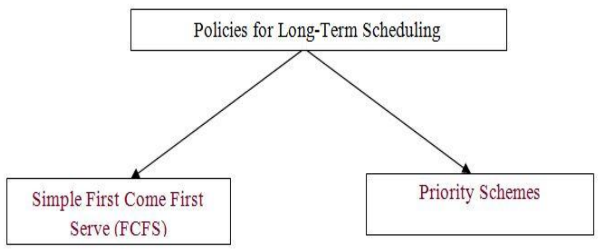
Policies for Long-Term Scheduling
Simple First Come First Serve (FCFS)
Priority Schemes

Figure 1: Long term scheduling policies

5.2 Midterm scheduling:

The scheduling of processes is mainly done based on the requirement of the resources. It is essentially concerned with memory management and often designed as a memory management subsystem of an operating system. It temporarily removes a process from the main memory which is of low priority or has been inactive for a long time. This is known as "swamping out" of a process. The scheduler may decide to swamp out the process which is page-faulting frequently or a process which is taking large amount of memory. Its efficient interaction with the short term scheduler is very essential for the performance of the systems with virtual memory.

5.3 Short term scheduling/CPU Scheduler:

It is concerned with the allocation of CPU time to meet the processes in order to meet some pre-defined objectives. It decides which of the in-memory process is executed following an interrupt or operating system call. This scheduler makes more frequent scheduling decisions than long-term and mid-term schedulers [5, 10] It can be preemptive or non-preemptive depending on the requirement. Mostly, it is written in assembler as it is a critical part of operating system

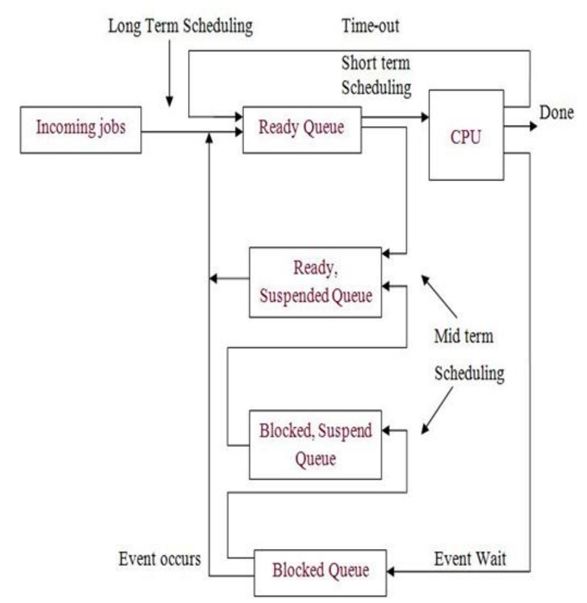

Figure 2 Queuing Diagram for scheduling

6 SCHEDULING PROBLEMS:

Consider we have m machines M_j(j=1,2,......m) and we have to process n jobs J_i(i=1,2,....n). Allocation of time intervals on machine is known as schedule and is represented by Gantt charts, which can be machine oriented or job oriented. [9]

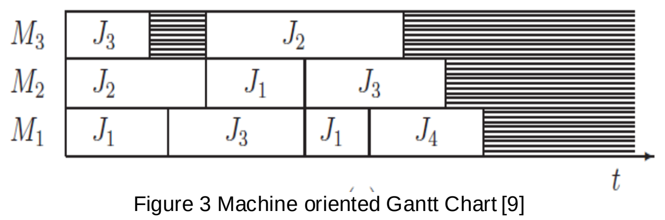

Figure 3 Machine oriented Gantt Chart [9]

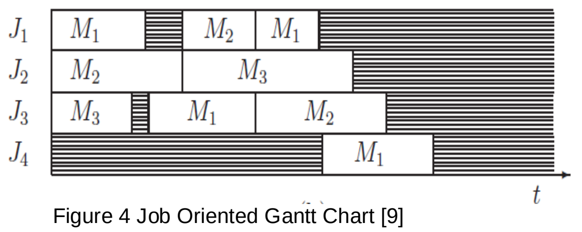

Figure 4 Job Oriented Gantt Chart [9]

Basic notations used in scheduling:

n_i: it represents the number of operations which is consisted in Job J_i, represented by $O_{i1}, O_{i2}, \ldots, O_{i,ni}$

p_{ij}: processing time requirement associated with operation O_{ij}

r_i: represents when the first operation of job J_i is available for processing, known as release date.

μ_{ij}: represents the set of machines associated with each operation O_{ij}. $\mu_{ij} \subseteq \{M_1, M_2, \ldots, M_m\}$. any operation can be processed on any machine. d_i: due date or deadline(in case of real time operations), time by which J_i should be ideally completed.

w_i: weight, representing the relative importance of J_i.

$f_i(t)$: non decreasing cost function, that measures the cost of completing J_i in time t. There are many classes of scheduling problems which are classified according to their complexity and specified in terms of classification which consists of three fields [10].

7 SCHEDULING ALGORITHMS:

It is a method by which resources are allocated to the processes such as bandwidth, processor time etc. It is usually done to distribute workload across multiple computers or processing units so as to achieve optimal resource utilization, maximize throughput, minimize response time and avoid overload [4]. The goal of scheduling algorithm is to fulfil the following criterion.

i. No Starvation:

This means that a particular process should not be held up indefinitely. There should be proper allocation of resources to every process so as to ensure that all processes get CPU time.

ii. Preemption in case of priority based algorithms

Scheduling algorithms have to ensure that there should be fairness in the preemption policy. It should make sure that high priority tasks should not hold low priority tasks indefinitely.

7.1Categorization of scheduling algorithms

Scheduling algorithms can be classified on the basis of the type of environment in which they are being used. This categorization is based on user's point of view. According to this, scheduling algorithms can be classified into three categories.

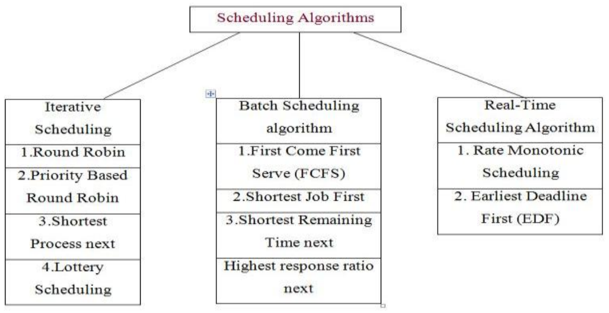
Scheduling Algorithms
Iterative Scheduling
1.Round Robin
2.Priority Based Round Robin
3.Shortest Process next
4.Lottery Scheduling
Batch Scheduling algorithm
1.First Come First Serve (FCFS)
2.Shortest Job First
3.Shortest Remaining Time next
Highest response ratio next
Real-Time Scheduling Algorithm
1. Rate Monotonic Scheduling
2. Earliest Deadline First (EDF)

Figure 5 Categorization of scheduling algorithms

7.2 First Come First Serve scheduling (FCFS):

It is the simplest scheduling algorithm in which processes are dispatched according to their arrival time on the ready queue. It is a non pre-emptive technique, so when processes get the CPU time, they are executed to completion. It is fair in the human sense of fairness but unfair in the sense that long jobs make short jobs wait or important jobs might get held up because of unimportant jobs. As the process is executed until it gets completed, there is no condition of starvation as long as process completes its execution.

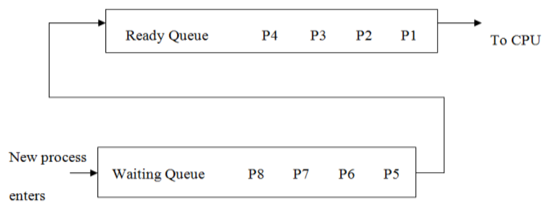

Figure 6 Scheduling processes using First Come First Serve (FCFS) technique

No prioritization of processes is there. It cannot guarantee good response time, so this technique is inappropriate for interactive systems. It is rarely used in modern operating systems but it is usually embedded within other schemes. [5]

7.3 Shortest Process Next:

In this scheduling policy, scheduler schedules the processes with the least estimated processing time to be next in queue. This technique is also known as Shortest-Job-First or shortest next-CPU-burst first. CPU burst is the time processor is being used by process before it is no longer ready or how long a process requires CPU between I/O waits. This policy considers that we know the next CPU burst of all the ready processes. The estimation of the length of next CPU burst is based on the recent CPU bursts. This technique is non-preemptive but pre-emption can be included with this technique and the resulting technique is known as Shortest Remaining Time next. The problem with shortest job first algorithm is that it cannot handle infinite loops and delivers poor performance if the task with short CPU burst comes after the process with longer burst which has started execution. Another problem associated with this technique is starvation i.e. processes with long burst times are indefinitely postponed from getting on the processor. In most of the cases, this technique is designed to achieve maximum throughput as short processes get most of the CPU time.

7.4 Scheduling Issues

While performing scheduling, there are multiple constraints which are to be met for accurate scheduling. There are issues relating to different tasks the system performs. The requirement of the system decides the scheduling algorithm design. We have to consider the application profile of the system; the level of scheduling required, the time at which a process is scheduled, the parameter to be optimized, the need of pre-emption etc. Some of the issues are classified below:

1. **Application Profile**
   It specifies the type of applications to be run on the system. A program alternates between CPU utilization and I/O. While performing scheduling we need to consider the bound of the programs executed on the system. It can be either compute-bound or I/O bound. For the I/O bound operations scheduler decides which processes should be scheduled while the currently running task is performing some I/O operation.

2. **Scheduling level**
   There can be only one process at a time which gets CPU time. So, the scheduler has to decide the process which is to be executed next. The decision is dependent on the choice of scheduling algorithm. Swapper is provided to decide about the processes which should reside in the memory. This type of scheduling is known as midterm scheduling.

3. **Time of schedule**
   It is the duty of scheduler to decide about the time at which schedule is started. This is done by the help of scheduling algorithm. The response of the system on receiving an interrupt, on creation/termination of a process, etc is determined by the scheduler. When there is a system call, it is the scheduler which decides which processes are to be put in ready queue.

4. **Pre-emptive or Non pre-emptive**
   The system can be interrupted during the execution of the process in case of arrival of a higher priority process. This type of system is known as pre-emptive system. If no interruption is there then the system is non pre-emptive system.

8. Conclusion and Future Scope:

The various objectives of my work have been met. Study of scheduling algorithms have been done and it has been observed that preemptive scheduling with dynamic priorities works very well in case of scheduling tasks on real time systems. From the comparison of real time scheduling algorithms, it is clear that earliest deadline first is the efficient scheduling algorithm if the CPU utilization is not more than 100%. For hard real time systems, calculations of probabilistic worst case execution time (WCET) analysis can be done. Implementation of scheduling algorithm on FPGA can be done for scheduling tasks with dynamic priorities and schedulability of the task can be checked.

REFERENCES

[1] Giorgio C. Buttazzo, "Hard Real Time Computing Systems: Predictable Scheduling Algorithms and Applications", Springer, Third edition.
[2] Eric Brewer, "Lottery Scheduling", Advanced Topics in Computer Systems, www.cs.berkeley.edu/~brewer/cs262/Lec-scheduling.pdf
[3] Franco Callari, "Types of Scheduling - Long Term and Medium Term Scheduling", http://www.cim.mcgill.ca/~franco/OpSys-304-427/lecture-notes/node38.html.
[4] Howard Hamilton, Kimberly Lemieux, Allan Tease, "Scheduling of Processes", http://www2.cs.uregina.ca/~hamilton/courses/330/notes/scheduling/scheduling.html
[5] "CPU Scheduling", http://www.os-concepts.thiyagaraaj.com/cpu-process-scheduling.
[6] Alan Burns and Sanjoy Baruah, "Sustainability in Real-time Scheduling", Journal of Computing Science and Engineering, Vol. 2, No. 1, March 2008, Pages 74-97.
[7] Daniel P. Bovet and Marco Cesati, "Understanding the Linux Kernel", O'Reilly Online Catalogue, October 2000.
[8] Hermann Kopetz, "Real-Time Systems: Design Principles for Distributed Embedded Applications", Springer, second edition.
[9] Peter Brucker, "Scheduling Algorithms", Springer, fifth edition.
[10] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan., "Optimization and approximation in deterministic sequencing and scheduling: A survey", Annals of Discrete Mathematics, 5:287–326, 1979.
[11] A. Burns, R.I. Davis, P. Wang, and F. Zhang " Partitioned EDF scheduling for multiprocessors using a C = D task splitting scheme", Real Time Systems, DOI: 10.1007/s11241-011-9126-9, Springer Science+Business Media, LLC 2011
[12] Wann-Yun Shieh, Bo-Wei Chen "Energy-Efficient Tasks scheduling Algorithm for Dual-core Real-time Systems", International Computer Symposium, December 2010, pp.568-575.
[13] Marko Bertogna and Sanjoy Baruah, "Limited Preemption EDF Scheduling of Sporadic Task Systems", IEEE Transactions on industrial informatics, vol. 6, no. 4, November 2010.

[14] Xian-Bo He, Gang-Yuan Zhang, Min Liu, Yu-Ping Zhao, Wei Li "A Fuzzy EDF Scheduling Algorithm Being Suitable for embedded Soft Real-time Systems in the Uncertain Environments", Advanced Computer Control (ICACC), 2nd International Conference, vol. 5, March 2010, pp. 583-587

[15] Fengxiang Zhang and Alan Burns, "Schedulability Analysis for Real-Time Systems with EDF Scheduling", IEEE transactions on computers, vol. 58, no. 9, September 2009, pp. 1250-1258

[16] Euiseong Seo Jinkyu Jeong, Seonyeong Park, and Joonwon Lee., "Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors", IEEE transactions on parallel and distributed systems, vol. 19, no. 11, November 2008, pp 1540-1552.

[17] Ming Xiong, Song Han, Kam-Yiu Lam and Deji Chen, "Deferrable Scheduling for Maintaining Real-Time Data Freshness: Algorithms, Analysis, and Results", IEEE transactions on computers, vol. 57, no. 7, July 2008, pp. 952-964.

[18] Sanjoy K. Baruah, and Joel Goossens, "Rate-Monotonic Scheduling on Uniform Multiprocessors", IEEE Transactions on Computers, Vol. 52, No. 7, July 2003, pp. 966-970.