Comparative Study of Scheduling Algorithms for Real Time Environment

Ishan Khera
Research Scholar
Thapar University, Patiala.

Ajay Kakkar
Asstt. Prof.
Thapar University, Patiala.

ABSTRACT:
Scheduling is a technique which makes an arrangement of performing certain tasks at specified period. The intervals between each function have been clearly defined by the algorithm to avoid any overlapping. The real time computing systems are those in which there are strict timing constraints that have to be met to get the correct output i.e. the output not only depend on the correctness of the outcome but also on the time at which results are produced. Real time systems are expected to change its state in real time even after the controlling processor has stopped its execution. The bound in which real time applications are needed to respond to the stimuli is known as deadline. In order to achieve optimized results in a real rime operations the scheduling techniques has been used. In the paper we classify the various scheduling techniques based on different parameters. Also techniques used for scheduling in real time environment are analyzed and comparison between different techniques have been done. The various issues have been presented on which there is still a need to work.

Key Words
Scheduling, Worst case execution time(WCET) power, real time operations RM algorithm, EDF, comparison of real time algorithms.

1. INTRODUCTION
Scheduling 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. It is done by a means of scheduler and dispatcher. 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. Deadline is the most important parameter of real time systems which is defined as the instant at which the results should be produced.
There can be three types of deadlines, which are mentioned below[3].
Soft Deadline: If the results produced after the deadline has passed and are still useful then this type of deadline is known as soft deadline. Reservation systems come under this category.
Firm deadline: This deadline is one in which the results produced after the deadline is missed is of no utility. Infrequent deadline misses are tolerable. These types of deadlines are used in systems which are performing some important operations.
Hard deadline: If catastrophe results on missing the deadline then this type of deadline is known as hard deadline. The systems which are performing critical applications like air traffic control come under this category.
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.[1]. Long-term scheduler is also known as Admission Scheduler
Midterm Scheduling
It is essentially concerned with memory management and often designed as a memory management subsystem of an operating system[1]. It temporarily removes a process from the main memory which is of low priority or has been inactive for a long time.

Short term Scheduling
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[1].
Schedulability Test: A test that determines whether the ready tasks can be scheduled in order to meet its specified deadlines. An optimal scheduler is one which can always find a feasible schedule whenever it exists[2]. There are three type of schedulability tests.
a. Sufficient Schedulability test
The positive outcome of this test guarantees that all the deadlines are always met.
b. Necessary Schedulability test
The failure of this test will definitely result in a deadline miss at some point of time.
c. Exact Schedulability test
A test which is sufficient and necessary is known as exact schedulability test.

2. LITERATURE SURVEY
The following sections show the work done by the various researchers in the field of scheduling for real time processors.
In 2008 Euiseong Seo et. al. [6] 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, S. Roman et. al. [7] presented Constant complexity scheduling for hardware multitasking in two dimensional reconfigurable field-programmable gate arrays which described the need of extending operating system to manage the FPGAs. An algorithm has been presented which decides the scheduling and placing of arrival tasks with real time constraints, like deadline, in FPGA device. The division of FPGA is done in four parts with different sizes and each

incoming task was placed in one of these partitins depending upon its size and parameters. It was also possible to make changes in the size of partition, number of partitions, queue selection policy etc. at the run time. The results proved that their four partition configuration is equally or more effective than FF algorithm. Even in the worst case their algorithm is better than FF, for eg., in case of pJPEG.

In 2008, Pravanjan Choudhary et. al. [8] presented hybrid scheduling of dynamic task graphs with selective duplication for multiprocessors under memory and time constraints which describes the scheduling methodology for the task graphs to embedded systems with multiple processors. Methodology has been presented in three phases which was designed for task graphs that were dynamic in nature due to the presence of conditional tasks and tasks those were unpredictable and bounded. The nodes were mapped in the first phase and the critical nodes were identified in the second phase. Those critical nodes were duplicated for the possible rescheduling at runtime. The third phase is the online phase which perform runtime scheduling. The experiments indicated that the methodology proposed is suitable for task graphs that have higher number of conditions, a higher parallelism, and a significant nondeterminism in the execution time of its nodes. It has been shown that computation time of the static phase and overhead increases with increase in number of nodes but still remained in acceptable limits. Overall the scheduling overhead stayed in microseconds which was very low as compared to conventional techniques.

In 2009, Enrico Bini et. al. [9] presented a response-time bound in fixed-priority scheduling with arbitrary deadlines in which they indentified the desirable properties of estimates of the exact response time. Repeated computation of the worst case response times slows down the system which is undesirable for real time applications. They proposed a technique which possessed the properties, those are, continuity with respect to system parameters, efficient computability and approximability for the estimation of worst case response time of sporadic task systems which are fixed priority scheduled on a preemptive uniprocessor. They have derived the continuous upper bound on the response times of task systems and the proved that the exact response time should be as large as this bound if the system is implemented on the processor which is at most only 50% as fast.

In 2009, Marko Bertogna et. al. [10] given the schedulability analysis of global scheduling algorithms on multiprocessor platforms in which they had addressed the problem of preemptive scheduling of periodic and sporadic task sets with constrained deadlines on a multiprocessor platform. They had assumed that the global work conserving scheduler with migration possibility of task from one processor to other was there. The analysis had been applied to fixed priority and Earliest deadline first scheduling techniques and the schedulability conditions thus derived were tightened which resulted in significant improvement in percentage of the accepted task sets. The schedulability algorithm presented could check whether a periodic or sporadic task set could be scheduled on a multiprocessor platform in polynomial or pseudo polynomial time. The iterative algorithm given by the procedure SCHEDULABILITYCHECK could detect the maximum number of schedulable task sets among all existing task sets.

In 2012, Wan Yeon Lee [11] proposed energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors which considered the processors contained more processing cores than running tasks and had dynamic voltage and frequency scaling capabilities. The energy saving techniques are introduced which were turning off the rarely used cores and exploiting the overabundant cores for executing the task in parallel with lower frequency. It had been verified if the techniques were supported then the problem of minimizing energy consumption of real time tasks was reduced to NP hard and deadlines were also met. A polynomial time scheduling scheme was also proposed that provided a near minimum energy feasible schedule. The evaluation results showed that the proposed scheme saved up to 64% of the energy consumed as compared to the other schemes which considered the execution of each task on a separate core.

3. OBJECTIVES OF SCHEDULING

From the work done by the various researchers in the field of real time scheduling; so far, it has been observed that

a. Scheduling should be done in order to guarantee the schedule of the processes fairly and throughput must be maximum.

b. Real time scheduling algorithms are always pre-emptive which can perform better if the pre-emption is limited.

c. Static priority scheduling algorithms are used for scheduling real time tasks for maximum CPU utilization but it can be increased more using dynamic priorities.

d. The schedulability of scheduling algorithm must be checked using schedulability tests.

4. SCHEDULING ALGORITHMS

Allocation of various resources like bandwidth, processor time etc are allocated to the processes using programs known as scheduling algorithms. Scheduling is done by distributing the workload among multiple computers or processing units in order to achieve optimal resource utilization, maximize throughput and minimize response time[4].

The goal of scheduling algorithm is to fulfil the following criterion.

a.Starvation should not be there which means a particular process should not be held indefinitely. Allocation of resources should be such that all the processes get proper CPU time in order to prevent starvation.

b.In case of priority based algorithms, there should be fairness in the pre-emption policy. Low priority tasks should not wait indefinitely because of higher priority tasks.

4.1 Classification of scheduling algorithms

Algorithms are classified on the basis of different parameters. The classification can be based on the type of application for which scheduling needs to be done, whether the tasks need to run time schedule or compile time schedule, whether the scheduling is done on central site or distributed sites, uniprocessor/multiprocessor scheduling.

4.1.1 Classification from user's point of view

From the user's point of view, scheduling algorithms are classified into 3 categories.

a. Iterative Scheduling

Scheduling of processes is done iteratively and the algorithms used for scheduling are known as iterative scheduling algorithms[2]. Round Robin, shortest process next, lottery scheduling etc are the examples of iterative scheduling algorithms

b.Batch Scheduling

Processes are queued together in a batch and scheduling is done in batches. Algorithms that are used to schedule the batches are known as batch scheduling algorithms. FCFS, Shortest remaining time next, highest response ratio next are the examples of batch scheduling algorithms.

c.Real time scheduling

Real time tasks are those in which the accuracy of the outcome not only depend on the correctness of result but also depend on the time at which the results are produced. Scheduling such tasks are done by real time scheduling algorithms[3]. Rate monotonic and Earliest Deadline First(EDF) are examples of real time scheduling algorithms.

4.1.2 Classification based on the time of schedule

This classification is done on the basis of time of scheduling the processes i.e. whether the processes are to be scheduled on the compile time or run time.

a.Static Scheduling

In this technique, scheduling decisions are made at compile time. For scheduling, complete prior knowledge of task-set characteristics is required. The system's behaviour with static scheduling is deterministic[4]. Rate monotonic scheduling is the example of static scheduling used for scheduling real time tasks.

b.Dynamic Scheduling

Scheduling decisions are made at run time by selecting one task out of the set of ready tasks[3,4]. Dynamic schedulers are flexible but also require run time in finding a substantial schedule. System's behaviour is non-deterministic. EDF is the example of dynamic priority scheduling algorithm used to schedule real time tasks.

4.1.3 Classification based on the site of scheduling

This classification is based on whether the processes are to be scheduled on a central site or on distributed sites.

a.Centralized Scheduling

All decisions are made at the central site[1]. The central scheduler in the distributed system is a critical point of failure. Updated load situation information is needed on all nodes which might lead to communication bottleneck.

b.Distributed scheduling

Scheduling of non-interactive processes or jobs in a network of computers comes under this category[2]. It refers to the chaining of different jobs into a coordinated workflow that spans several computers.

4.1.4 Classification based on pre-emption

Pre-emption means prediction of higher priority task. Depending upon, whether pre-emption is allowed or not, scheduling algorithms can be classified into two categories.

a.Pre-emptive Scheduling

If the system can be interrupted during the execution of the process on the arrival of higher priority task then this type of system is known as pre-emptive system and scheduling algorithms used to schedule such systems are known as pre-emptive scheduling algorithms[2,4]. All real time scheduling algorithms are examples of pre-emptive scheduling algorithms.

b.Non Pre-emptive Scheduling

If no interruption is allowed during the execution of process then scheduler is known as non pre-emptive scheduler. First Come First Serve(FCFS) scheduler is non pre-emptive scheduler.

4.1.5 Classification based on the number of processes to be scheduled

This classification is done considering whether the scheduling is done on single processor or multiple processors.

a.Uniprocessor Scheduling

If the scheduling is done on a single processor then it is known as uniprocessor scheduling[3]. Round Robin, RM scheduling etc are the examples of uniprocessor scheduling algorithms.

b.Multiprocessor Scheduling

If number of events occurring close together are high then we have to increase number of processors in the system. Such system is known as multiprocessor systems and scheduling techniques reuired to schedule a task on such system are known as multiprocessor scheduling algorithm[1,4]. Global scheduling algorithms and partitioning scheduling algorithms fall under this category.

5. DYNAMIC SCHEDULING ALGORITHMS WITH STATIC PRIORITIES

5.1 Rate Monotonic Scheduling algorithm (RMS)

It is a dynamic pre-emptive algorithm for scheduling set of independent hard real time tasks. This was published in 1973 by Liu and Layland[5]. The algorithm was based on static task priorities. The assumptions made about the task set are mentioned below[3,4].

1. The request for all the task sets, for which hard deadlines should be met, are periodic.
2. All tasks are independent of each other. No precedence constraints or mutual exclusion constraints exist between any pair of tasks.
3. The deadline interval of every task is equal to its period.
4. The required maximum computation time is known beforehand and is constant.
5. Time required for context switching can be ignored.
6. Sum of utilization factors of n tasks with period period p is given by

$U=\Sigma(c_i/p_i) \leq n(2^{1/n} - 1)$.

As n approaches infinity, term $n(2^{1/n} - 1)$ reaches ln 2 (about 0.7).

The task priorities are assigned on the basis of their periods. The task with shortest period gets the highest priority and the task with longest period gets lowest priority. If all the assumptions stated above are satisfied then this algorithms guarantees that all the tasks will meet their deadlines. The algorithm is optimal for single processor systems.

5.2 Deadline Monotonic (DM)

This technique is an extension of Rate Monotonic scheduling algorithm. This is first proposed in 1982 by Leung and Whiteland. This is also fully pre-emptive technique used for scheduling tasks with static priorities[3]. The third assumption mentioned in rate monotonic technique that says the deadline interval of every task is same and equal to its period; has been relaxed. The tasks have constrained deadlines i.e. relative deadlines can be less than or equal to its period.

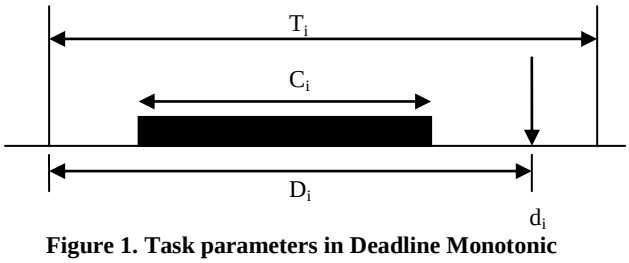

Figure 1. Task parameters in Deadline Monotonic scheduling [4]

C_i: Worst case computation time
D_i: Relative Deadline
T_i: Period of the task

Each task is assigned a fixed priority inversely proportional to its relative deadline D_i. So, at any instant task with the

shortest deadline is executed. As relative deadlines are constant, DM is a static priority assignment technique.

6. DYNAMIC SCHEDULING ALGORITHMS WITH DYNAMIC PRIORITIES

6.1 Earliest Deadline First algorithm (EDF)

It is the optimal dynamic pre-emptive algorithm for a single processor systems which are based on dynamic priorities. The tasks with the earliest deadline is given the highest priority[2]. That is why this algorithm is also known as deadline driven scheduling algorithm(DDS). The jobs in the task set are put in the ready queue on the basis of their priorities. The priorities of all the jobs in the ready queue are fixed. So, this algorithm is a job level fixed-priority algorithm. A task set can be scheduled by this algorithm if the utilization factor, $U \leq 1$.

6.2 Least Laxity algorithm (LL)

It is an optimal algorithm in single processor systems. Laxity of the task is defined as a difference between deadline interval and maximum computation. Laxity of a task i, $l_i = d_i - c_i$. d_i is the deadline interval of task i, which is the duration between the deadline of the task and the task request instant. c_i is the computation time of task i. The assumptions about the task sets are same as that of EDF algorithm. The priorities are assigned on the basis of laxity[3]. The task with longest laxity gets the lowest priority. This may cause frequent switching between the tasks and thus increase the system overhead.

7. COMPARISON OF REAL TIME SCHEDULING ALGORITHMS

	Rate Monotonic(RM)	Deadline Monotonic (DM)	Earliest Deadline First (EDF)	Least Laxity First (LL)
Implementation	Simplest	Simple	Difficult	Difficult
Processor Utilization	Less	More as compared to RM	Full Utilization	Full Utilization
Priority Assignment	Static	Static	Dynamic	Dynamic
Scheduling criterion	Task Period	Relative Deadline	Deadline	Laxity i.e. d_i-c_i
Jitter Control	Only for highest priority task	Only for highest priority task	Inefficient in overloaded systems	Inefficient in overloaded systems

8. CONCLUSION AND FUTURE SCOPE

From the comparative study it can be concluded that real time systems have became an inevitable part of our lives and their accurate performance has been a challenge. Scheduling a real time systems is done by static priority scheduling algorithms. Although dynamic priority scheduling algorithms can perform better than static priority in terms of CPU utilization, but it increases the overhead on the system. So, dynamic priority scheduling algorithms are not available in commercial real time systems. The various objectives of scheduling have been discussed. Study of scheduling algorithms have been done and it has been observed that pre-emptive 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, schedulability analysis can be done and calculations of probabilistic Worst Case Execution Time (WCET) analysis can also 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.

9. REFERENCES

[1] Daniel P. Bovet and Marco Cesati, "Understanding the Linux Kernel", O'Reilly Online Catalogue, October 2000.

[2] Hermann Kopetz, "Real-Time Systems: Design Principles for Distributed Embedded Applications", Springer, second edition.

[3] Peter Brucker, "Scheduling Algorithms", Springer, fifth edition.

[4] Giorgio C. Buttazzo, "Hard Real Time Computing Systems: Predictable Scheduling Algorithms and Applications", Springer, Third edition.

[5] C.L. Liu and James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment", Journal of the Association of Computing Machinery, Vol. 20, No. 1, January 1973, pp. 46-61.

[6] 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.

[7] S. Roman, H. Mecha, D. Mozos and J. Septien, " Constant complexity scheduling for hardware multitasking in two dimensional reconfigurable field-programmable gate arrays", IET Computer Digit. Tech., 2008, Vol. 2, No. 6, pp. 401–412.

[8] Pravanjan Choudhury, Rajeev Kumar and P.P. Chakrabarti, "Hybrid Scheduling of Dynamic Task Graphs with Selective Duplication for Multiprocessors under Memory and Time Constraints" IEEE Transactions On Parallel And Distributed Systems, Vol. 19, No. 7, July 2008, pp. 967-980.

[9] Enrico Bini, Thi Huyen Chau Nguyen, Pascal Richard, and Sanjoy K. Baruah, "A Response-Time Bound in Fixed-Priority Scheduling with Arbitrary Deadlines", IEEE Transactions On Computers, Vol. 58, No. 2, February 2009, pp. 279-286.

[10] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari, "Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms", IEEE Transactions On Parallel And Distributed Systems, Vol. 20, No. 4, April 2009, pp. 553-566.

[11] Wan Yeon Lee, "Energy-Efficient Scheduling of Periodic Real-Time Tasks on Lightly Loaded Multicore Processors", IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 3, March 2012, pp. 530-537