Research Article Impact Factor: 0.621 ISSN: 2319-507X
PM Kanhekar, IJPRET, 2014; Volume 2 (8): 279-288 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

OVERVIEW OF REAL TIME SHEDULING TECHNIQUES

P. M. KANHEKAR[1], PROF. S. A. BHURA[2]

1. ME Student, Department of Computer Sc. & Engg., Babasaheb Naik College of Engg., Pusad.
2. Associate Professor, Department of Computer Sc. & Engg., Babasaheb Naik College of Engg., Pusad.

Accepted Date: 15/02/2014 ; Published Date: 01/04/2014

Abstract: This paper focus on overview of clock driven, priority-driven, weighted round-robin approaches to scheduling. An algorithm for scheduling hard real time jobs is optimal if it can produce a feasible schedule as long as feasible schedules of the given jobs exist i.e. when system is not overloaded. Another important concept is predictability of the timing behavior of jobs. The execution behavior of system is predictable if system exhibits no anomalies. We can conclude that the jobs in a predictable system can always meet their deadlines if the jobs meet their deadline according to the maximal schedule of the system i.e., when every job in the system executes for as long as its maximal execution time.

Keywords: Real time system, Scheduling techniques, Clock–driven scheduling, Priority-driven scheduling, Weighted round robin scheduling.

PAPER-QR CODE

Corresponding Author: MR. P. M. KANHEKAR

Access Online On:

www.ijpret.com

How to Cite This Article:

PM Kanhekar, IJPRET, 2014; Volume 2 (8): 279-288

INTRODUCTION

Real-time computing plays a crucial role in our society since an increasing number of complex systems rely, in part or completely, on computer control. Examples of applications that require real-time computing include nuclear power plants, railway switching systems, automotive and avionic systems, air traffic control, telecommunications, robotics, and military systems. In the last several years, real-time computing has been required in new applications areas, such as medical equipments, consumer electronics, multimedia systems, flight simulation systems, virtual reality, and interactive games Real-time systems are computerized systems with timing constraints [2]. Real-time systems can be classified as hard real-time systems and soft real-time systems. In hard real-time systems, the consequences of missing a task deadline may be catastrophic. In soft real-time systems, the consequences of missing a deadline are relatively milder. Examples of hard real-time systems are space applications, Fly-by-wire aircraft, radar for tracking missiles, etc. Examples of soft real-time systems are on-line transaction used in airline reservation systems, multimedia systems, etc [1],[3]. This thesis

Deals with scheduling of periodic tasks in hard real-time systems. Applications of many hard real-time systems are often modeled using recurrent tasks. For example, real-time tasks in many control and monitoring Applications are implemented as recurrent periodic tasks. This is because periodic execution of recurrent tasks is well understood and predictable. The most relevant real-time task scheduling concepts in this thesis are: periodic task system, ready (active) task, task priority, preemptive scheduling algorithm, feasibility condition of a scheduling algorithm, offline and online scheduling[1] [3].

II. Real-Time Characteristics

Real-time systems often are comprised of a controlling system, controlled system and environment. Controlling system: acquires information about environment using sensors and controls the environment with actuators. Timing constraints derived from physical impact of controlling systems activities. Hard and soft constraints. Periodic Tasks: Time-driven recuring at regular intervals. Aperiodic: event-driven.

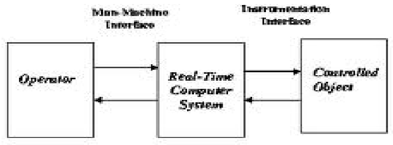


Fig 1 Real time system

A formal verification which guarantees all deadlines in a real-time system would be the best. This verification is called feasibility test. Three different kinds of tests are available:-1.Exact tests with long execution times or simple Models [7],[8],[9].

2.Fast sufficient tests which fail to accept feasible task Sets, especially those with high utilizations [10], [11].

3.Approximations, which are allowing an adjustment of Performance and acceptance rate [12], [13].

III. Scheduler Characteristics

A scheduler provides an algorithm or policy for ordering the execution of the outstanding processes on the processor according to some pre-defined criteria. In a conventional Multitasking operating system, processes are interleaved with higher importance (or priority) processes receiving preference. Little or no account is taken of deadlines. This is clearly inadequate for real time systems. These systems require scheduling policies that reflect the timeliness constraints of real-time processes. Schedulers produce a schedule for a given set of processes. If a process set can be scheduled to meet given pre-conditions the process set is termed feasible. A typical pre-condition for hard real-time periodic processes is that they should always meet their deadlines. An optimal scheduler is able to produce a feasible schedule for all feasible process sets conforming to a given precondition. For a particular process set an optimal schedule is the best possible schedule according to some pre-defined criteria. Typically a scheduler is optimal if it can schedule all process sets that other schedules can[1].

IV. Real-time systems and Scheduling techniques

Real– time system are defined as those systems in which the correctness of the system does not depend only on the logical results of computations but also on the time at which the results are produced [1]. Real-time systems have well defined, fixed time constraints i.e., processing must be completed within the defined constraints otherwise the system will fail. There are two main types of real-time systems: Hard Real-Time System, Firm or Soft Real-Time System. In Hard Real-Time System requires that fixed deadlines must be met otherwise disastrous situation may arise whereas in Soft Real-Time System, missing an occasional deadline is undesirable, but nevertheless tolerable. System in which performance is degraded but not destroyed by failure to meet response time constraints is called soft real time systems. The objective of a real-time task scheduler is to guarantee the deadline of tasks in the system as much as possible when we consider soft real-time system. Mostly all the real-time systems in existence use preemption and multitasking. We observe that the choice of an operating system is important in designing a real time system. Designing a real time systems involves choice for proper language, task portioning and merging and assigning priorities using a real time scheduler to manage response time. The depending upon scheduling objectives parallelism and communication may be balanced. The designer of scheduling policy must be determine critical tasks and assign them high priorities however, care must be taken avoid starvation, which occurs when high priority tasks are always ready to run. Real-time scheduling techniques can be largely divided into two categories: Static and Dynamic. Static algorithms assign priorities at design time. All assigned priorities remain constant for the lifetime of a task. Dynamic algorithms assign priorities at runtime, based on execution parameters of tasks.

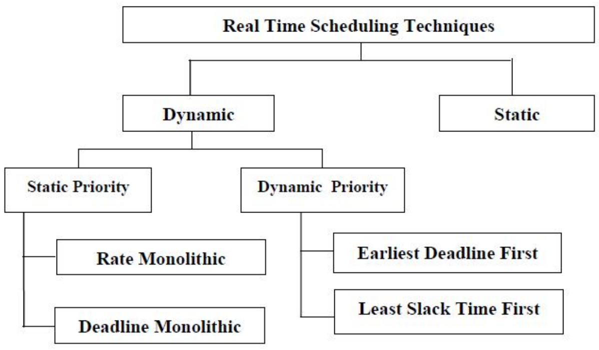


Figure 2. Types of Real Time Task Scheduling Technique

Dynamic scheduling can be either with static priority or dynamic priority. R.M (Rate Monolithic) and DM(Deadline Monolithic) are examples of dynamic scheduling with static priority. EDF(Earliest Deadline First) and LST (Least Slack Time First) are examples of dynamic scheduling with dynamic priority. EDF and LST algorithms are optimal under the condition that the jobs are preemptive, there is only one processor and the processor is not overloaded . But the limitation of these algorithms is, their performance decreases exponentially if the system becomes slightly overloaded[1], [2][4]

A .Clock-Driven scheduling

Clock-driven scheduling applicable to deterministic systems. . A restricted periodic task model. The parameters of all periodic tasks are known a priori. For each mode of operation, system has fixed number, n, periodic tasks. For task Ti each job Ji, k is ready for execution at its release time ri, k and is released pi units of time after the previous job in Ti such that ri, k = ri,k-1 + pi.

Variations in the inter-release times of jobs in a periodic task are negligible. A periodic job may exist. Assume that the system maintains a single queue for periodic jobs. Whenever the processor is available for a periodic job, the job at the head of this queue is executed. There are no sporadic jobs Recall: sporadic jobs have hard deadlines, a periodic jobs do not. Decisions about what jobs execute when are made at specific time instants. These instants are chosen before the system begins execution. Usually regularly spaced, implemented using a periodic timer interrupt. Scheduler awakes after each interrupt, schedules the job to execute for the next period, then blocks itself until the next interrupt E.g. 1.the helicopter example with an interrupt every 1/180th of a second 2.the furnace control example, with an interrupt every 100ms. Typically in clock-driven systems: All parameters of the real-time jobs are fixed and known . A schedule of the jobs is computed off-line and is stored for use at runtime; as a result, scheduling overhead at run-time can be minimized .Simple and straight-forward, not flexible. Clock-driven Scheduling [14].

Advantages

I. Conceptual simplicity

1) Ability to consider complex dependencies, communication delays, and resource contention among jobs when constructing the static schedule, guaranteeing absence of deadlocks and unpredictable delays.

2) Entire schedule is captured in a static table

3) Different operating modes can be represented by different tables

4) No concurrency control or synchronization required

5) If completion time jitter requirements exist, can be captured in the schedule [1],[14].

II. When workload is mostly periodic and the schedule is cyclic, timing constraints can be checked and enforced at each frame boundary.

III. Choice of frame size can minimize context switching and communication overheads.

IV. Relatively easy to validate, test and certify.

Disadvantages

I. Inflexible

1) Pre-compilation of knowledge into scheduling tables means that if anything changes materially, have to redo the table generation

2) Best suited for systems which are rarely modified once built?

II. Other disadvantages:

1) Release times of all jobs must be fixed

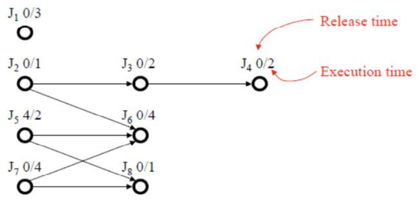
2) All possible combinations of periodic tasks that can execute at the same time must be known a priori, so that the combined schedule can be precomputed

3) The treatment of a periodic job is very primitive

4) Unlikely to yield acceptable response times if a significant amount of soft real time computation exists [1].

B. Priority-driven scheduling

Priority-driven algorithm differ from each other in how priorities assigned to jobs .We classify the algorithm into two types Fixed-priority and Dynamics-priority .A fixed-priority algorithm assign same priority to all the jobs in each task i.e. priority of periodic task are fixed relative to the others Assign priorities to jobs, based on some algorithm .Make scheduling decisions based on the priorities, when events such as releases and job completions occur. Priority scheduling algorithms are event-driven .Jobs are placed in one or more queues; at each event, the ready job with the highest priority is executed .The assignment of jobs to priority queues, along with rules such a whether preemption is allowed, completely defines a priority scheduling algorithm. Priority-driven algorithms make locally optimal decisions about which job to run. Locally optimal scheduling decisions are often not globally optimal. Priority-driven algorithms never intentionally leave any resource idle .Leaving a resource idle is not locally optimal Example: Priority-Driven Scheduling Consider the following task: Jobs J1, J2,..., J8, where Ji had higher priority than J-k if



i<k Jobs are scheduled on two processors P1 and P2.Jobs communicate via shared memory, so communication cost is negligible. The schedulers keep one common priority queue of ready jobs. All jobs are preempted able; scheduling decisions are made whenever some job becomes ready for execution or a job completes. The ability to preempt lower priority jobs slowed down the overall completion of the task .This is not a general rule, but shows that priority scheduling results can be non-intuitive. Different priority scheduling algorithms can have very different properties. Tracing execution of jobs using tables is an effective way to demonstrate

correctness for systems with periodic tasks and fixed timing constraints, execution times, resource usage Show that the system enters a repeating pattern of execution, and each hyper-period of that pattern meets all deadlines .Proof by exhaustive simulation. Provided the system has a manageably small number of jobs. Most scheduling algorithms used in non real-time systems are   priority-driven

1. First-In-First-Out
2. Last-In-First-Out
3. Shortest-Execution-Time-First
4. Longest-Execution-Time-First

Real-time priority scheduling assigns priorities based on deadline or some other timing constraint: Earliest deadline first, Least slack time first [1][14]

a) . Earliest Deadline First Approach

A preemptive priority-based scheduling scheme is assumed. The algorithm used for scheduling is: the process with the (current) closest deadline is assigned the highest Priority in the system and therefore executes. The schedulability constraint is given thus: Hence, 100% processor utilization is possible. This is a sufficient and necessary condition for schedulability. for an arbitrary process set in which process timing constraint is relaxed to allow deadlines not equal to periods, condition (2) is necessary but not sufficient. Liu and Layland state that given the constraints in the simple model, "the deadline driven scheduling algorithm is optimum in the sense that if a set of processes can be scheduled by any algorithm, it can be scheduled by the deadline driven algorithm.”. Treatment of a non-preemptable process scheme is given by Jeffay32 in which non-preempt able processes are shown to be scheduled by the earliest deadline heuristic if they can be scheduled by At any instant it executes the task with the earliest absolute deadline among all the ready tasks. A set of n independent tasks with arbitrary arrival times, the EDF algorithm is optimal with respect to minimizing the maximum lateness. Assign priority to jobs based on deadline. This algorithm requires the pre-knowledge of deadlines. Earlier the deadline, the priority. The choice of relative deadline arises naturally from throughput consideration .The job in each period completes before the next period starts so there is no backlog of jobs. As observation regarding EDF is “The schedulable utilization Uedf(n) of the algorithm for n independent ,preempt able periodic tasks with deadline equal to or larger than their periods is equal 1”and also” A system of independent , preempt able periodic tasks with relative deadlines longer than there can be feasibly scheduled on a processor as long as the total utilization is equal to or less than 1[1]”.

b).Schedulability Test For EDF Algorithm

For the purpose of validating that the given application system can indeed meet all its hard deadlines when scheduled according to the chosen scheduling algorithm. A Schedulability test is efficient; it can be used as an on-line acceptance test. Checking whether a set of periodic tasks meet all their deadlines is special case of the validation problem that can be stated as follows:

A. The period pi execution time ei, and relative deadlines Di of every task Ti in a system T = [T1, T2….Tn] of independent periodic tasks.

B. A priority driven algorithm used to schedule the tasks in T preemptively on one processor

$$\sum_{k=1}^{n} (ek/\min(Dk,Pk)) <= 1$$

We check whether the inequality is satisfied. We call inequality the schedulability condition of the EDF algorithm .If it is get satisfied then only the process will follow the EDF algorithm. Else the conclusion we draw from this fact depends on the relative deadlines of the task. If Dk>=pk for all k from 1 to n then above equation reduces to U<=1, which necessary and sufficient condition for a system to be feasible; if Dk<pk for some k Eq.is only a sufficient condition; hence we can only say that the system may not be schedulable when the condition is not satisfied [1].

C. Weighted Round Robin Schedule

Regular round-robin scheduling is commonly used for scheduling time-shared applications . Every job joins a FIFO queue when it is ready for execution When the scheduler runs, it schedules the job at the head of the queue to execute for at most one time slice Sometimes called a quantum – typically O(tens of ms) If the job has no completed by the end of its quantum, it is preempted and Placed at the end of the queue when there are n ready jobs in the queue, each job gets one slice every n time slices (n time slices is called a round). Only limited use in real-time systems. Weighted round robin (WRR) scheduling discipline has been implemented in a broad range of mission critical computing and communication systems. In a WRR scheduler, tasks are performed in a cyclic order, in which the time a task can execute within each round is proportional to the weight assigned to it. Weighted round robin schedulers have two major

Advantages:

1. Ability to improve the system robustness, given their guarantee of a minimum service rate for each task. For weighted round robin schedulers, the maximum amount of service every task

Can receive in each round is upper-bounded by its allocation. As such, no task can consume more service than what has been assigned.

2. Ability for implementation in a distributed fashion. A WRR scheduler can be easily realized in a distributed environment through a logic ring like the timed token protocol.

By passing a token along the ring, nodes can access resource in turn based on their weight assignments. It has been proven in that with proper weight assignment; weighted round robin schedulers can provide deadline guarantees for real-time computing systems. Yet a major challenge is how to devise a low complexity schedulability test, which would guarantee the deadline requirements, achieve high level of resource utilization, and is applicable to a broad range of system configurations. According to a Schedulability test can be direct or indirect. A direct schedulability test explicitly calculates the precise worst-case delay of each task in order to determine the permissibility of a new task. Despite its accuracy, this type of test has high run-time cost in calculating the worst case task delays, and therefore may not be suitable for on-line admission control. In contrast, an indirect schedulability test would use one or more system indicators to determine the schedulability of a new task without computing the worst case task delays directly. The utilization based schedulability test is the most common indirect schedulability test, in which a new task can be admitted only if the total system utilization (as the system indicator) is lower than a pre-determined bound. Utilization bound based schedulability test is highly desirable for large, complex systems, because of its extreme efficiency and the ability to provide a safe operation margin by setting the system utilization bound to a value lower than the proven utilization bound [1],[5],[14].

V. Conclusion

This paper focus on auxiliary of scheduling techniques Priority-driven scheduling, Clock-Driven scheduling, Weighted Round Robin Schedule. Every techniques have some advantages and some disadvantages It concentrate on simplity then it get disadvantage on techniques so for overcome this it derived another techniques . more focus on time constraint and should execute within the deadline .

REFERENCE

1. Jane W.S. Liu "*Real-Time Systems"* Pearson Education, India,pp. 121 & 26, 2001

2. Buttazzo, G.C Hard Real-Time Computing System Predictable scheduling algorithm and application 2011.

3. Risat Mahmud Pathan. "*Scheduling Algorithms For Fault-Tolerant Real-Time Systems"* Copyright c Risat Mahmud Pathan, 2010. Technical Report No. 65L ISSN 1652-876X Department of Computer Science and Engineering Dependable Real-Time Systems Group

4. N.Audsley A. Burns "*Real-time system scheduling"* Department of Computer Science, University of York, UK.

5. M.Kaladevi, M.C.A., M.Phil. And Dr.S.Sathiyabama. "*A Comparative Study of Scheduling Algorithms for Real Time Task"* M.Sc.,M.Phil.,Ph.D,2International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010

6. Jianjia Wu, Jyh-Charn Liu, and Wei Zhao. "*Systems Schedulability Bound of Weighted Round Robin Schedulers for Hard Real-Time"* Department of Computer Science, Texas A&M University {jianjiaw, liu, zhao}@cs.tamu.edu

7. S. Baruah, S. Funk, and J. Goossens , "*Robustness Results Concerning EDF Scheduling upon Uniform Multiprocessors*",IEEE Transcation on computers, Vol. 52, No.9 pp. 1185-1195 September 2003

8. C.M. Krishna and Shin K.G. "*Real-Time Systems*" TataMcGrawiHill, 1997.

9. S. Baruah, D. Chen, S. Gorinsky, "*A. Mok. Generalized Multiframe Tasks. The International Journal of Time-Critical Computing Systems",* 17, 5-22, 1999.

10. S. Baruah, A. Mok, L. Rosier. "*Preemptive Scheduling Hard-Real-Time Sporadic Tasks on One Processor. Proceedings of the Real- Time Systems Symposium"*, 182-190, 1990.

11. K. Gresser. Echtzeitnachweis Ereignisgesteuerter Realzeitsysteme. Dissertation (in german), VDI Verlag, Düsseldorf, 10(286), 1993.

12. M. Devi. "An *Improved Schedulability Test for Uniprocessor Periodic Task Systems. Proceedings of the 15th Euromicro Conference on Real- Time System"*, 2003

13. C. Liu, J. Layland. "*Scheduling Algorithms for Multiprogramming in Hard Real-Time Environments".* Journal of the ACM, 20(1), 46-61, 1973.

14. *"Module Overview of Real Time System Scheduling".*