International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Analysis of Rate Monotonic Scheduling Algorithm on Multicore Systems

Manish S. Raut[1], M. B. Narnaware[2]

[1]Department of Information Technology Walchand College of Engineering, Sangli

[2]Professor, Department of Information Technology Walchand College of Engineering, Sangli

Abstract: ***Real Time Systems are valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time. An appropriate scheduling algorithm with a corresponding scheduliability analysis is used to ensure a system is predictable. The scheduling algorithm is of paramount importance in a real-time system to ensure desired and predictable behavior of the system. Rate monotonic scheduling algorithm (RMS) is one of the most commonly used fixed priority scheduling algorithm in Real Time Systems.***

Keywords: Scheduling, Rate Monotonic Scheduling (RMS), Real-time Systems, Multi-core Systems

1. Introduction

Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced. A Real-time-system is a computer system in which the key aspect of the system is to perform tasks on time, not finishing too early nor too late. A classic example is that of the air-bag in a car, it is of great importance that the bag initiates neither too soon nor too late in orders to be of aid and not be potentially harmful [3].

In Real-time system, a valid schedule is a feasible schedule if every job completes its deadline (or, in general meets its timing constraints). The setof job is schedulable according to scheduling algorithm if when using the algorithm the scheduler always produces a feasible schedule [7].

The scheduling algorithm is of paramount importance in a real-time system to ensure desired and predictable behavior of the system. Within computer science real-time systems are an important while often less known branch. Cars, planes and entertainment systems are just some devices in which real- time systems reside, governing the workings of that device while we do not consider that such a system exists within the chosen device [1].

2. Literature Survey

Real-time scheduling techniques can be divided into two categories: Static and Dynamic. EDF (Earliest Deadline First) and LST (Least Slack Time First) are examples of dynamic scheduling with dynamic priority. On the other hand Rate Monotonic, Deadline Monotonic algorithms are examples of static scheduling with static priority [6].

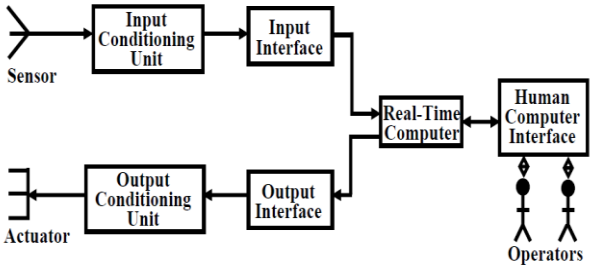

Figure 1: Real Time System.

Although the Earliest Deadline First (EDF) algorithm is optimal for preemptive real-time scheduling on uniprocessor, fixed priority scheduling such as the Rate Monotonic (RM) algorithm has been more widely used in real-time systems because it is very easy to implement [1]. In [6], RMalgorithm is proven to be optimal among all fixed priority scheduling methods when tasks are scheduled independently and preemptively on a uniprocessor system. A task is schedulable if and only if the worst case response time of the task is shorter than its period.

Periodic jobs are often scheduled preemptively on uniprocessor systems using a class of algorithms known as priority-driven algorithms. A priority-driven algorithm is one that never leaves the processor idle intentionally when there are requests ready for execution.

Such an algorithm can be implemented by assigning priorities to requests; at each instant the request with the highest priority among all ready requests is executed. Well-known examples are the earliest-deadline first algorithm and the rate-monotonic algorithm .The former assigns priorities to requests dynamically on the basis of their deadlines, the earlier the deadline of a request, the higher its priority. It is known to be optimal. The latter assigns priorities statically to jobs (and, hence, to individual requests in them) on the basis of their periods, shorter the period of a job, the higher its priority [2].

3. Related Works

Task scheduling of real-time systems on multi-core architecture.[3].The Real time systems and the RMS algorithm is briefly described by the W S Liu.[7]. The scheduliability test and task model for RMS are briefly described here for the given set of tasks.

Rate monotonic scheduling algorithm can be implemented on multi-core embedded systems to schedule task using open MP programming model ParaRms [1], The solution proposed in this paper significantly enhances the performance of the real time embedded system.

4. Rate Monotonic Scheduling

Rate monotonic scheduling is optimal static priority scheduling algorithm. The static priority algorithm is the one which allocates priorities before execution of processes [2]. The algorithm is optimal in the sense that the set of tasks that cannot meet the deadlines assigned to them, RMS [4] cannot schedule them. The processor with the least period has the first priority [8].

Formulas check for scheduliability with rate monotonic scheduling of task set. The purpose of a real-time scheduling algorithm is to ensure that critical timing constraints, such as deadlines and response time are met [2]. For a process the response time is defined as the time at which it finishes its execution [3].

We had already pointed out that RMA is an important event-driven scheduling algorithm. This is a static priority algorithm and is extensively used in practical applications [5]. RMA assigns priorities to tasks based on their rates of occurrence. The lower the occurrence rate of a task, the lower is the priority assigned to it. A task having the highest occurrence rate (lowest period) is accorded the highest priority. RMA has been proved to be the optimal static priority real-time task scheduling algorithm. In RMA, the priority of a task is directly proportional to its rate (or, inversely proportional to its period). That is, the priority of any task Ti is computed as: priority = k / pi, where pi is the period of the task Ti and k is a constant. Using this simple expression, plots of priority values of tasks under RMA for tasks of different periods can be easily obtained. These plots are shown in Fig. 2(a) Fig. 2(b). It can be observed from these figures that the priority of a task increases linearly with the arrival rate of the task and inversely with its period [5].

The underlying theory of RMS is known as Rate Monotonic Analysis (RMA).It has the following assumptions:
A1: All processor are allocated the single CPU based on their periods.
A2: Context switching time is not considered.
A3: No data dependencies are there among the processes.
A4: The execution time for each of the process is constant.
A5: Deadlines are considered to be at the end of periods.
A6: The process in ready state and having highest priority is considered for execution.

Given certain information about a particular set of tasks, under rate monotonic conditions, one can evaluate certain tests to understand whether or not those tasks can all meet their deadlines in a real time system. Because these values are known at design time and are monotonic, any analysis and scheduling can be done statically.

Static scheduling is one advantage that the industry has a strong preference for in hard real time applications. Given the computation time, Ci, and period, Ti, for task i, its CPU utilization can be calculated with the following equation:

$$U_i = \frac{C_i}{T_i} \quad (1)$$

The utilization bound (UB) test allows scheduliability analysis by comparing the calculated utilization for a set of tasks and comparing that total to the theoretical utilization for that number of tasks.

$$\frac{C_1}{T_1} + ... + \frac{C_n}{T_n} <= U(n) = n(n^{1/2} - 1) \quad (2)$$

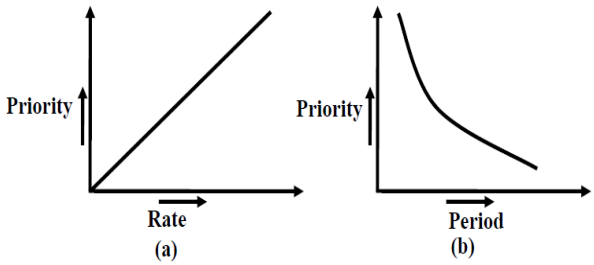

Figure 2: Priority Assignment to Tasks in RMA

If this equality is satisfied, all of the tasks will always meet their deadlines. But what in case where this equality is not being satisfied, the task will not get completed and will miss its deadline.

Let's take an example: we have three tasks as T1,T2 and T3 with their respective periods as 4, 5, 7 and their respective Execution time as 1, 2, 2. In the Fig(3) we can see that the periods have given the priority according to their deadlines and the task with the least period has given the highest priority. So they are sorted in increasing order oh their periods. At one point the task T3 with the period 7 and execution time 2 will miss its deadline as shown in Fig(3). This analysis for the above example has been done on unicore platform, therefore by using multicore systems we can assign the tasks T1, T2 and T3 to the different cores, and can schedule them without missing of the deadline..

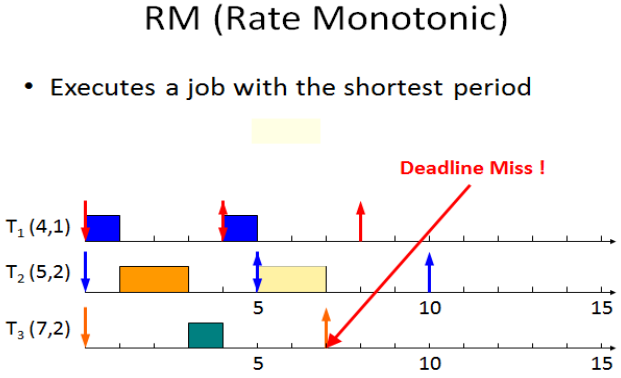

Figure 3: Rate monotonic: missing of deadline

5. RMS Scheduling on Multicore Systems

A solution with which the RMS algorithm can be applied in parallel has been given in [1]. The algorithm is executed on multicore processes so that many tasks can be scheduled at a given instance of time. A core comprises of various threads (thread pool) so these threads, whenever idle, can pick up a task and execute it. Implementation of the solution is done in OpenMP. ParaRMS Algorithm enhances the performance of the Embedded System to a significant level of optimization using OpenMP. Tasks are executed by balancing their loads on thethreads and also they can be dynamically scheduled as some orthe other threads that are free can execute the tasks appearing dynamically [1].

ParaRMS very efficiently synchronizes the processes among cores and has very less overhead time on symmetric multi-core embedded processor which is analyzed and the experimental results are in fever of this claim. This solution is strongly scalable because the number of processor cores increase the algorithm will take less time process the same task-set also it can schedule more number of processes. Hence increasing number of cores in the system increases its capacity to schedule more number of tasks [1].

6. Conclusion and Future Work

If we really want a scalable system where no (or minimum) deadlines should be missed then the focus should be more on minimizing the missing of deadline, as it's a real time system we don't want to execute that task before the required time, as RMS Algorithm has polynomial time complexity, it may not affect in excess for the timing behavior of the tasks. Also by analyzing the scheduliability of the given real time system for uniprocessor, we can conclude that if all jobs of all tasks are achieving their deadline then, the given real time system is reliable on uniprocessor system. Else if some jobs are missing the deadline then we conclude that the given real time system is not reliable on uni-core computing system and multi-core system will be required for the of the system.

In the future scope we are proposing a multicore model where we can schedule these tasks on multicore model where we can allocate different cores for the different tasks and can complete those tasks without missing the deadline.

References

[1] Dashora R, Bajaj P, Dube A, and Narayanamoorthy M, "ParaRMS algorithm: Aparallel implementation of rate monotonic scheduling algorithm using OpenMP",IEEE Conference on Advances in Electrical Engineering (ICAEE), vol.78, no. 6, pp.855-860, 2014.
[2] Shih, W. K, Liu J. Liu c, "Modified Rate- Monotonic Algorithm for SchedulingPeriodic Jobs with Deferred Deadlines", IEEE Transactions on Software Engineering,vol. 19, pp. 1171-1179, 1993.
[3] Pengliu Tan, "Task Scheduling of Real-time Systems on Multi-Core Architectures",School of Computing Nanchang Hangkong University Nanchang, China.
[4] Rohit Chandra, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan and Je_McDonald, Parallel Programming In OpenMP, Academic press, A Harcourt Scienceand Technology Company, USA, 2001.
[5] Embedded System Software. Version 2 IIT, Kharagpur [Online]. Available: http://www.nptel.ac.in/courses/108105057/Pdf/Lesson-30.pdf
[6] C. Liu and J. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, J. ACM, vol. 20, no. 1, pp. 4661, 1973.
[7] Jane W.S. Liu Real Time Systems, Pearson edition.
[8] Moonju Park and Heemin ParkAn, "Efficient Test Method for Rate Monotonic Schedulability", IEEE Trans. on Computer, vol. 63, no. 5, 2014.