International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 7, July-2014, pp: (306-309), Impact Factor: 1.252, Available online at: www.erpublications.com

Scheduling for Overload in Real-Time Systems

Apurva Shah
Department of Computer Science and Engineering, Faculty of Technology & Engineering,
The Maharaja Sayajirao University of Baroda, Vadodara, India

Abstract: Scheduling of real-time systems in overloaded conditions is a challenging problem. No on-line scheduling algorithm operating in an uniprocessor environment can guarantee to obtain a useful processor utilization greater than 0.25 under conditions of overload. The issue of improving overload performance in environments where there is a soft deadline has been addressed in this paper. The need of improving Earliest Deadline First algorithm has been discussed and modified algorithms are simulated on uniprocessor, multiprocessor and distributed environment.

Keywords: Real-time systems, multiprocessor, distributed, uniprocessor, Earliest deadline first, scheduling.

Introduction

Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical results of computations but also on the time at which the results are produced [1]. The objective of the real-time system is to meet these deadlines, that is to process tasks before their deadlines expire. Therefore, in contrast to conventional computer systems where the goal usually is to minimize task response times, the emphasis here is to on satisfying the timing constraints of tasks.

A multiprocessor system is tightly coupled so that global status and workload information on all processors can be kept current at a low cost. The system has shared memory and generally uses centralized scheduler. If system uses separate scheduler for each processor, the decisions and actions of the schedulers of all the processors are coherent.

Multiprocessor systems are divided in two basic types, homogeneous and heterogeneous. In homogeneous system, processors can be used interchangeably and in contrast, heterogeneous processors cannot be used interchangeably. Homogeneous processors can be subdivided in two types: identical and uniform. In identical processors, it is assumed that all processors are equally powerful whereas uniform multiprocessor machine is characterized by a speed [2]. There are two main strategies when dealing with the problem of multiprocessor scheduling: partitioning strategy and global strategy [3]. In a partitioning strategy, once a task is allocated to a processor, all of its instances are executed exclusively on that processor. In a global strategy, any instance of a task can be executed on any processor, or even be preempted and moved to a different processor before it is completed [4].

Real-time scheduling techniques can be broadly divided into two categories Static and Dynamic. Static algorithms assign all priorities at design time and it remains constant for the lifetime of the task. Dynamic algorithms assign priority at runtime, based on execution parameters of tasks. EDF (Earliest Deadline First) and LLF (Least Laxity First) algorithms are proved optimal under the condition that tasks are preemptive, there is only one processor and it is not overloaded ([5], [6]). EDF is appropriate algorithm to use for on-line scheduling on uniform multiprocessors [7]. However, for multiprocessor systems no online scheduling algorithm can be optimal ([8], [9]. Many practical instances of multiprocessor real-time system are NP-complete, i.e. it is believed that there is no optimal polynomial-time algorithm for them ([1], [10]).

The scheduling is considered as on-line, if scheduler makes scheduling decision without knowledge about the task that will be released in the future. On-line scheduling algorithms cannot work efficiently in overloaded conditions.

Researchers have proved that, for single processor system, the competitive factor of an on-line scheduling algorithm is at most equal to 0.25 when the system is highly overloaded and its value can't be more than 0.385 when the system is slightly overloaded. They have further derived the upper limit of competitive factor for dual processor system and it is 0.5 during overloading condition ([11], [12]). It has been also proved that multiprocessor system is schedulable with guarantee when utilization is below 0.37482 [13].

Modified EDF Algorithm

To achieve the goal of meeting all task deadlines, the designers of safety-critical real-time systems attempt to anticipate every eventuality and incorporate it into the design of the system. Such system will never lose deadline in ideal

conditions and behave as per expectation of the system designer. In practical situations, unanticipated emergency conditions may occur and it may require to handle overload conditions.

Classical EDF algorithm is used extensively for real-time systems. This algorithm which processes tasks in deadline order is optimal under normal conditions and it meets all the deadlines. But during overload condition it performs worse than even random algorithms. Therefore, there is need to improve EDF algorithm which can resist overload conditions too [14]. Following modification in classical EDF have been suggested to meet such requirement.

O_EDF Scheduling Algorithm :

During underloaded condition the algorithm works same as EDF algorithm i.e. priority of the jobs will be decided dynamically depending on their deadline. During overloaded condition, those jobs are discarded whose expected time for execution is more than their relative deadline. The improved algorithm has been applied for uniprocessor real-time systems. Also, with necessary changes in the algorithms, it has been applied for tightly coupled and loosely coupled multiprocessor environment [16], [17]. The discarding of tasks in overloaded conditions can be controlled using static priorities, which will be used only in overloaded conditions. The higher priority task can be allowed to execute in any condition.

System and Environment

The system assumed here is:

- Uniprocessor Real-Time System
- Tightly coupled multiprocessor real-time system
- Loosely coupled multiprocessor real-time system

The tightly coupled multiprocessor real-time system is assumed with shared memory and soft timing constraints. The processors assumed are homogeneous and identical. The system has adopted global strategy along with centralized scheduler. The system permit task preemption and inter-processor migrations (i.e. a job executing on a processor may be interrupted at any instant and its execution resumed later on the same or a different processor with no cost or penalty).

The loosely coupled multiprocessor system assumed here is client/server distributed system without shared memory. Each client makes scheduling and resource access control decisions independently. The clients assumed are homogeneous and can be used interchangeably for different kinds of tasks. Each client can execute at most one task at a time and task preemption is allowed.

Simulation Method

A reasonable way to measure the performance of a scheduling algorithm during an overload is by the amount of work the scheduler can feasibly schedule according to the algorithm. The larger this amount the better the algorithm. In real-time systems, deadline meeting is most important and we are interested in finding whether the task is meeting the deadline. Therefore the most appropriate performance metric is the Success Ratio and defined as [15],

$$SR = \frac{Number\ of\ tasks\ successfully\ scheduled}{Total\ number\ of\ tasks\ arrived} \quad (1)$$

Results

EDF and O_EDF algorithm have been applied for all uniprocessor real-time systems and results are shown in Table 1 and Figure 1. As we can see that both proposed algorithms give value of SR 100% for underloaded conditions. Therefore, we can conclude that they are optimal for underloaded condition when system is with single processor. But as the load increases, EDF is not able to sustain. O_EDF performs quite well in overloaded conditions too [16].

Figure 2 shows the results with number of processors are three and as mentioned in case 2, the processors are tightly coupled. Results show that both algorithms perform quite well when the system is not overloaded. During overloaded condition, SR of conventional EDF fall down rapidly but O_EDF algorithm works better [17].

Figure 3 shows the results for loosely coupled environment. O_EDF algorithm works fine for loosely coupled environment also where EDF performs poor.

Table 1: Uniprocessor System During Underloaded Conditions

Load	SR%	
	EDF	O_EDF
0.50	100	100
0.60	100	100
0.70	100	100
0.80	100	100
0.90	100	100
1.00	100	100

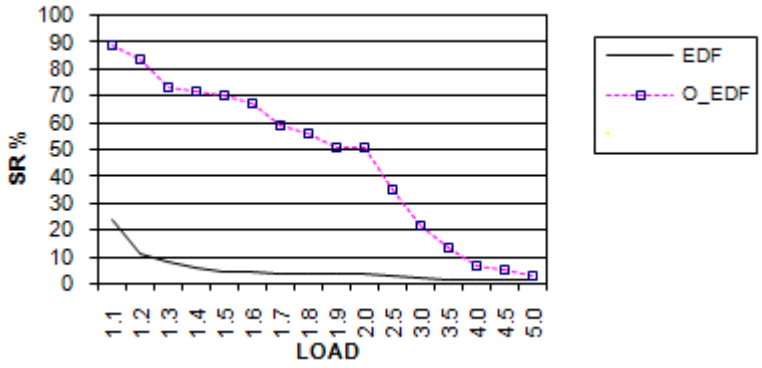

Figure 1: Uniprocessor environment

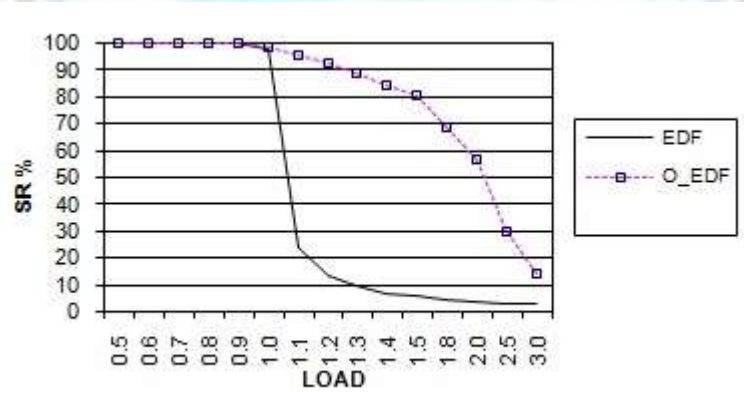

Figure 2: Tightly coupled multiprocessor with number of processors are Five

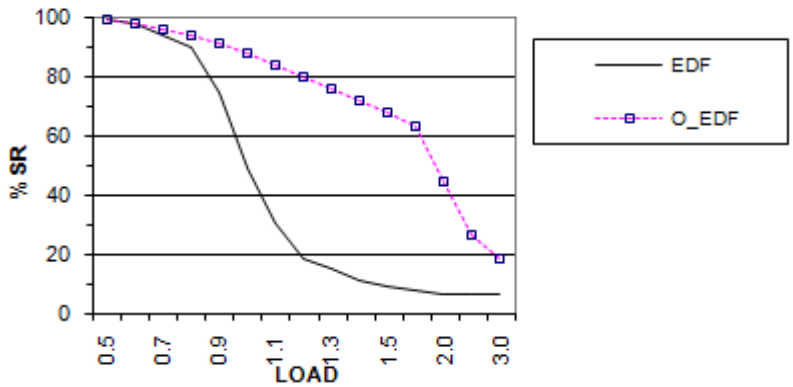

Figure 3: Loosely coupled multiprocessor with number of processors are Five

Conclusions

EDF gives optimal results but performs poor during overloaded conditions. With a minor modification in classical EDF, it improves the performance of the algorithm in uniprocessor and multiprocessor environments. The modified algorithm tested in all different environments and proven its efficiency.

References

[1]. K. Ramamritham and J. A. Stankovik, "Scheduling Algorithms and Operating Support for Real-Time Systems", Proc. of the IEEE, vol 82(1), pp. 55-67, 1994.

[2]. S. Baruah , S. Funk and J. Goossens, "Robustness results concerning EDF scheduling upon uniform multiprocessors", IEEE Tran on Computers, vol.52(9) , pp. 1185-1195, 2003.

[3]. Y. Oh , S. H. Son, "Allocating ¯xed priority periodic tasks on multiprocessors system, Real-Time Systems", vol. 9(3), pp. 207-239, 1995.

[4]. J. M. Lopez, J. L. Diaz and D. F. Garcia, "Mnimum and maximum utilization bounds for multiprocessor rate monotonic scheduling", IEEE Tran on Parallel and Distributed Computing, vol. 15(7) , pp. 642-653, 2004.

[5]. M. Dertouzos , K. Ogata, "Control Robotics: The procedural control of physical process", Proc. IFIP Congress, 1974.

[6]. A.K. Mok, "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment", Ph.D.Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.

[7]. S. H. Funk, J. Goossens and S. Baruah, "On-line scheduling on uniform multiprocessors", IEEE Real-Time Systems Symposium, pp. 183-192, 2001.

[8]. M. Dertouzos and A.K. Mok , "Multiprocessor on-line scheduling of hard-real-tme tasks", IEEE Transaction on Software Engineering, vol 15 (12), pp. 1497-1506 1989.

[9]. K. Hong and J. Leung , "On-line scheduling of real-time tasks", Proc. of Real-Time Systems Symposium, pp. 244-250, 1988.

[10]. J. D. Ulman, "Polynomial complete scheduling problems", Operating Sys Review, vol. 7, Oct 1973.

[11]. J.W.S. Liu, Real-Time Systems, Pearson Education, India, 2001.

[12]. S. Baruah S., G. Koren, B. Mishra , A. Raghunath , L. Roiser and D. Shasha, "On-line Scheduling in the presence of overload", In FOCS, pp. 100-110, 1991.

[13]. L. Lundberg, "Analyzing fixed-priority global multiprocessor scheduling", Proc. of Eighth IEEE Real-Time and Embedded Technology and Applicatons Symposium, 2002.

[14]. S.K. Baruah and J. R. Harista, "Scheduling for Overload in Real-Time Systems", IEEE Transactions on Computers, vol. 46 (9), Sep. 1997.

[15]. K. Ramamrithm, J.A. Stankovic and P.F. Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems", IEEE Transaction on Parallel and Distributed Systems, vol. 1(2), pp. 184-194, 1990.

[16]. K. Kotecha and A. Shah, "Efficient dynamic scheduling algorithms for real-time operating systems", International Confrenence on High Performance Computing, Networking and Communication Systems, pp. 21-25, July 2008.

[17]. A. Shah and K. Kotecha, "Efficient scheduling algorithms for real-time distributed systems", International Conference on Parallel, Distributed and Grid Computing, pp. 44-48, Oct. 2010.