Efficient Scheduling Algorithms for Real-Time Distributed Systems

Apurva Shah
G H Patel College of Engg & Tech.
Vallabh Vidyanagar, INDIA
shahapoorva@ymail.com

Ketan Kotecha
Nirma Institute of Technology
Ahmedabad, INDIA
drketankotecha@gmail.com

Abstract

Dynamic scheduling has been always a challenging problem for real-time distributed systems. EDF (Earliest Deadline First) algorithm has been proved to be optimal scheduling algorithm for single processor real-time systems and it performs well for distributed systems also. In this paper, we have proposed scheduling algorithms for client/server distributed system with soft timing constraints. The algorithms are proposed with modifications in conventional EDF algorithm.

The proposed algorithms are simulated; results are obtained and measured in terms of SR (Success Ratio) & ECU (Effective CPU Utilization). Finally the results are compared with EDF algorithm in the same environment. It has been observed that the proposed algorithms ares equally efficient during underloaded conditions and they perform much better during overloaded conditions.

Keywords: real-time systems, scheduling, distributed systems, EDF

1. Introduction

Real-time system is required to complete its work and deliver its services on a timely basis. The results of real-time systems are judged based on the time at which the results are produced in addition to the logical results of computations [1]. Priority based scheduling algorithms have widely applied in real-time systems and scheduling decisions are made immediately upon job releases and completions. These algorithms include Earliest deadline first (EDF) [2], Least Laxity First (LLF) [3], Highest Value First (HVF) [4], Highest Value Density First (HVDF) [5] etc. In these algorithms, priorities of tasks are all calculated based on some characteristics of tasks, such as dead line, laxity time, criticalness, etc. RM and DM algorithms are fixed priority algorithms and priority is decided off-line. In EDF and LLF algorithm priority changes as time changes and therefore, called dynamic algorithms.

However it is insufficient if the priority of a task is only determined by using a characteristic ([7], [8]). For example, EDF policy assigns the highest priority to the task with the earliest deadline, and LLF policy assigns the highest priority to the task with the shortest slack time. It is not most critical whether the deadline or the slack time is shorter or not. Especially, in overload situations, EDF or LLF algorithm will result in rapidly decreasing of the system performance, even bringing on the domino effect [5].

In client/server distributed systems, the server is often the bottleneck. For example, the growth rate of traffic at the World Wide Web servers has increased exponentially [9]. The same problem is affected to digital library systems also. Improving the server performance is thus crucial in improving the overall performance of distributed information systems. Load balancing or load sharing- these two are the main approaches for scheduling in distributed systems. In load balancing or load sharing, server has to play vital role and this leads to the problem as mentioned above.

2. System and Task Model

The system assumed here is client/server distributed system which is basically loosely coupled homogeneous multiprocessor system. The system is real-time system with soft timing constraints. Each client makes scheduling 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.

The system knows about the deadline and required computation time of the task when the task is released. The task set is assumed to be preemptive.

978-1-4244-7674-9/10/$26.00 ©2010 IEEE

In soft real-time systems, each task has a positive value. The goal of the system is to obtain as much value as possible. If a task succeeds, then the system acquires its value. If a task fails, then the system gains less value from the task ([10], [11]). In a special case of soft real-time systems, called a firm real-time system, there is no value for a task that has missed its deadline, but there is no catastrophe either ([12], [13]). Here, the proposed algorithm is assuming the system as firm real-time system. The value of the task has been taken same as its computation time required.

3. Overall Scheduling Strategy

In client/server distributed systems, improving the server performance is crucial in improving the overall performance of distributed information systems. Considering this criteria, we have proposed the scheduling algorithm for client while the server is transferring the different tasks to different clients one-by-one without considering load on the client or any other criteria. It is the responsibility of the client to achieve the deadline for different tasks assigned to it.

EDF algorithm performs well when the system is not overloaded but its performance decreases exponentially when system becomes slightly overloaded. To overcome the limitation of EDF algorithm in overloaded condition, we have applied some modifications in conventional EDF algorithm and proposed the new algorithms – named D_O_EDF and D_R_EDF.

3.1 D_O_EDF Scheduling Algorithm

During overloaded conditions, EDF is not able to perform well. The main reason for that is, it gives priority to those jobs which have missed the deadline. In real-time systems, those jobs are not useful which can not be completed in time. It is almost meaningless to run those jobs, which have already missed the deadline or expected to miss the deadline. The D_O_EDF algorithm suggests discarding this type of jobs.

Some of the jobs might be with soft timing constraints and they might not be useless even though after missing the deadline. Therefore, D_O_EDF algorithm is proposed with following modifications in conventional EDF algorithm.

- For this algorithm, the jobs are given static priority 0 and 1. This priority will be used only in overloaded conditions.
- During underloaded condition the algorithm works same as EDF algorithm, i.e. priority of the jobs will be decided dynamically depending on its deadline.
- During overloaded condition, those jobs are discarded whose expected time for execution is more than their respective deadline and their static priority is 0.
- Jobs with soft timing constraints are expected with static priority 1. This type of jobs are not discarded and allowed to execute even though they have missed the deadline or expected to miss the deadline.

3.2 D_R_EDF Scheduling Algorithm

EDF is dynamic priority algorithm which assigns priority based on deadline of the task and it is optimal for single processor during underloaded condition. Limitation of any dynamic (or on-line) algorithm is that they are not working efficiently in overloaded condition. RM is a static algorithm which assigns priorities to tasks based on their periods. Limitation of any static algorithm is that they are not as efficient as dynamic algorithm in underloaded conditions [6] but they are performing well during overload compared to dynamic algorithms. Therefore, the following new algorithm is proposed, which is combination of both static and dynamic approach with automatic switching among them.

- During underloaded condition the algorithm works same as EDF algorithm, i.e priority of the tasks will be decided dynamically depending on their deadline.
- During overloaded condition it works same as RM algorithm i.e. priority of the tasks will be decided depending on their period.
- When two continuous jobs miss the deadline, it will be identified as overloaded condition and the algorithm will switch for RM algorithm. After 5 jobs have continuously achieved the deadline, again the algorithm will shift back to EDF algorithm considering that overloaded condition is disappeared.
- During underloaded condition, aperiodic tasks will get priority according to their deadline as scheduling is done by EDF algorithm. But during overloaded condition, as the scheduling of the jobs is done by RM algorithm, the aperiodic jobs will get least priority and they will run only when no periodic tasks are waiting.

4. Simulation Method

We have simulated EDF algorithm and the proposed algorithm and results are produced. We have considered periodic tasks for taking the results as load of the system can be measured accurately for periodic tasks. For periodic tasks, load of the system can be defined as summation of ratio of required computation time and period of each task ([10], [14]). For taking result at each load value, we have generated 200 task sets each one containing 3 to 26 tasks. The results for 15 different values of load are taken ($0.5 \leq \text{load} \leq 3$) and tested on more than 36,000 task sets.

The system is said to be overloaded when even a clairvoyant scheduler cannot feasibly schedule the tasks offered to the scheduler [14]. 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. Because of this, we have considered following two as our main performance criteria:

1) 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] ,

$$SR = \frac{Number\ of\ tasks\ successfully\ scheduled}{Total\ number\ of\ tasks\ \ arrived}$$

2) Effective CPU Utilization (ECU) which can be defined as,

$$ECU = \sum_{i \in R} \frac{V_i}{T}$$

where,

- V is value of a task and,
  - Value of a task = Computation time of a task, if the task completes within its deadline.
  - Value of a task = 0, if the task fails to meet the deadline.
- R is the set of tasks, which are scheduled successfully i.e. completed within their deadline.
- T is total time of scheduling.

Competitive factor can also be considered as an important performance measuring criteria. An on-line scheduler has a competitive factor C_f if and only if the value of the schedule of any finite sequence of tasks produced by the algorithm is at least C_f times the value of the schedule of the tasks produced by an optimal clairvoyant algorithm [14]. Since maximum value obtained by a clairvoyant scheduling algorithm is a hard problem, we have instead used a rather simplistic upper bound on this maximum value, which is obtained by summing up the value of all tasks [16]. Therefore, we have considered value of ECU for clairvoyant scheduler is 100%.

Finally, the results are obtained and compared with EDF algorithm in the same environment and shown in Fig. 4 to Fig. 6.

5. Results

Figure 1 shows the results achieved by the proposed algorithms and EDF algorithm when number of processors is 2. We can observe that %SR and %ECU of EDF fall down rapidly but the proposed algorithms work better.

We can observe that in underloaded condition, EDF performs well and the proposed algorithm is also equally well. But as the load is increasing slightly beyond 1.00 to 1.10, performance of EDF is decreased drastically, while the proposed algorithms sustain well.

From the values of %ECU and considering maximum value for clairvoyant scheduler, we find that competitive factor of the algorithms discussed. Competitive factor of the D_O_EDF algorithm is more than 0.47 and 0.33 when load values are 1.50 and 2.00. Competitive factor of the D_R_EDF algorithm is at least 0.46 and 0.45 for the same load values. In the same condition, competitive factor of EDF is less than 0.06 and 0.04.

Figure 2 and 3 demonstrate the same when numbers of processors are 3 and 5.

6. Conclusions

The algorithms discussed in this paper are for scheduling of client/server distributed system with soft timing constraints. As the proposed algorithms work well in both underloaded and overloaded conditions, they can be used when future workload of the system is unpredictable.

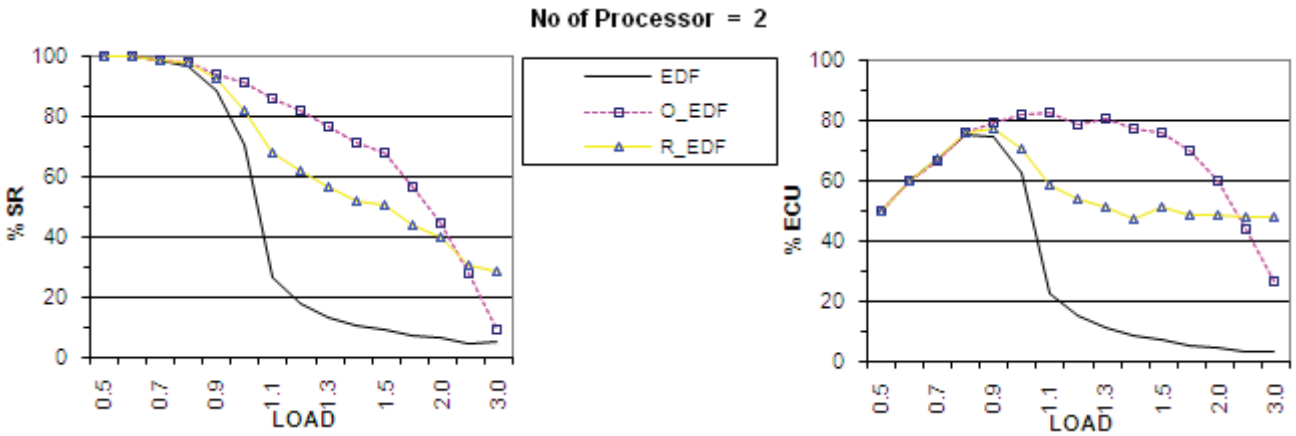

Figure1. Load Vs %SR and Load Vs % ECU when number of processor = 2

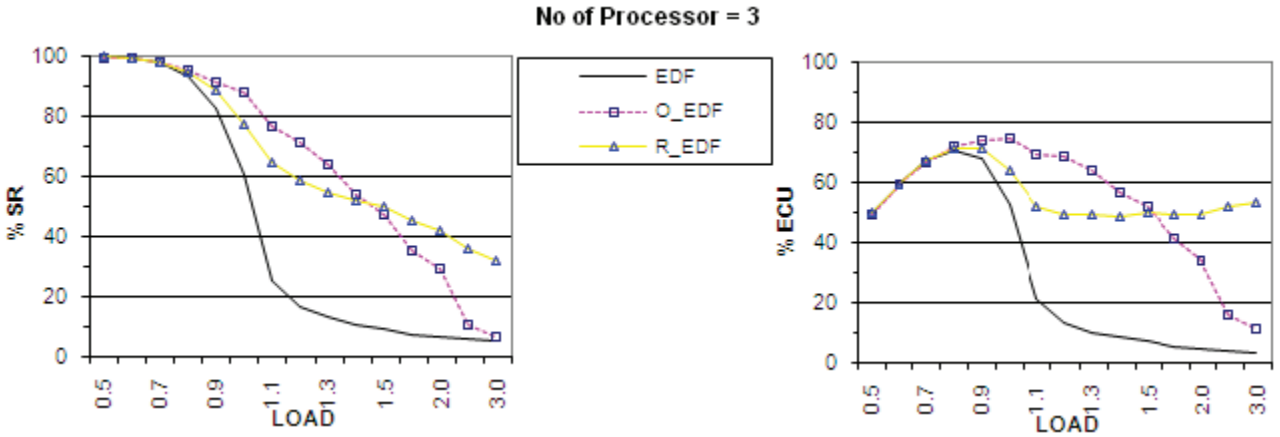

Figure 2. Load Vs %SR and Load Vs % ECU when number of processor = 3

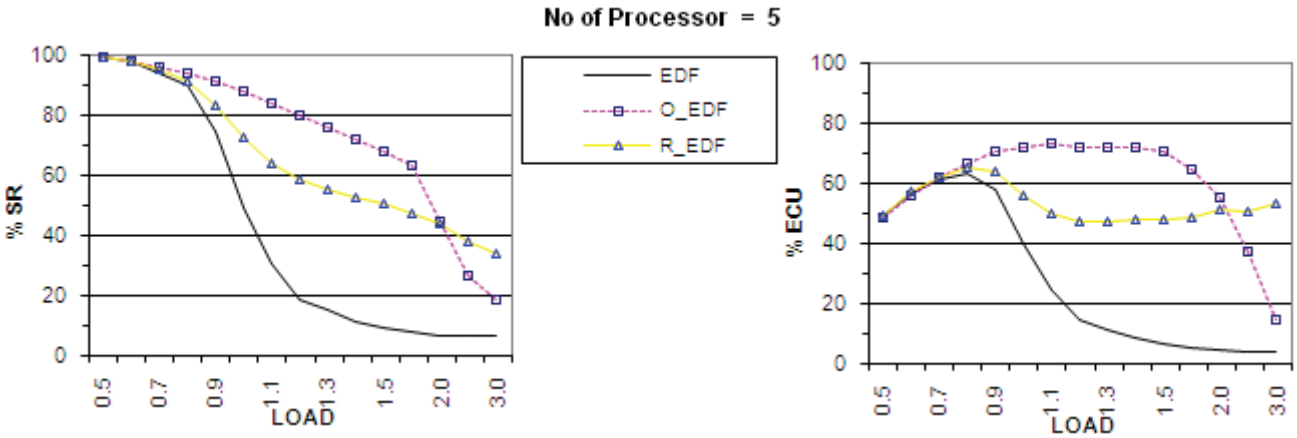

Figure 3. Load Vs %SR and Load Vs % ECU when number of processor = 5

References

[1] K. Ramamritham and J.A. Stankovik, "Scheduling algorithms and operating support for real-time systems", *In proc. of the IEEE*,vol 82, January 1994.

[2] Dertouzos M. and Ogata K, "Control Robotics: The procedural control of physical process", *In proceeding of IFIP Congress,* 1974.

[3] Mok A, "Fundamental design problems of distributed systems for the hard-real time environment", *PhD thesis,* MIT, Cambridge, 1983.

[4] Jensen E D, Locke C.D. and Toduda H, "A time driven scheduling model for real-time operating system", *In proceedings of RTSS*, 1985*,* pp. 112-122.

[5] Buttazzo G.,.Spuri M. and Sensini F., "Value Vs. Deadline scheduling in overload conditions", *In proceeding of RTSS*, 1995*,* pp. 90-95.

[6] Liu C.L and Layland. "Scheduling algorithms for multiprogramming in a hard real-time environment". *Journal of ACM.* Vo. 20, 1973, pp. 46-61.

[7] A. Burns, D. Prasa, A. Bondavalli, et el., "The Meaning and role of value in scheduling flexible real-time systems", *Journal of Systems Architecture*, vol.46, 2000, pp. 305-325.

[8] S.R.Biyabani, J.A. Stankovic and K. Ramamrithm, "The integration of deadline and criticalness in hard real-time scheduling", *In proc. of RTSS,* 1988, pp. 152-160.

[9] L. Yongcheng and C. Roy, "A Dynamic priority based scheduling mehod in distributed systems", *In Proc. of the Int. Conference on Parallel and Distributed Processing Techniques and Applications,* 1995, pp. 177-186.

[10] K.Kotecha and A. Shah, "Efficient dynamic scheduling algorithms for real-time multiprocessor systems", I*n proceedings of HPCNCS 08, Fl,* 2008, pp. 21-25.

[11] C.D. Locke., "Best Effort Decision Making for Real-Time Scheduling", *Ph.d.thesis*, Computer Science Department Carnegie-Mellon University, 1986.

[12] G. Koren and D.Shasha, "D^{Over}: An optimal on-line scheduling algorithm for overloaded real-time systems*", SIAM J of Computing*, vol 24, 1995, pp. 318–339.

[13] K.Kotecha and A. Shah, "Adaptive scheduling algorithms for real-time operating systems", I*n proceedings of CEC 08(under WCCI08),HongKongl,* 2008, pp. 2109-2112.

[14] J.W.S. Liu, Real-time systems, Pearson Education, India, 2001.

[15] K.Ramamritham ,Stankovik J.A. and Shiah P.F, "Efficient scheduling algorithms for real-time multiprocessor systems", *IEEE Transaction on Parallel and Distributed Systems*,1(2), April1990.

[16] S Baruah, G Koren, B. Mishra, et al.,"On-line scheduling in the presence of overload", *in FOCS,* 1991, pp. 100-110.