International Journal of Computer Applications (0975 – 8887)
Volume 31– No.6, October 2011

Enhancement in LLF Real-Time dynamic Scheduling Algorithm using conventional RM algorithm

Prem Sindhi
M. Tech Computer Science
SSSIST, Sehore

Ravindra K. Gupta
Assistant Professor
Computer Science Department
SSSIST, Sehore

ABSTRACT

To achieve timing requirements in real-time systems scheduling is very crucial. Essentiality of any Real-time system task is decided on the basis of deadline, slack time, or period of its occurrence. Earliest deadline first (EDF), least-laxity-first (LLF) also known as smallest slack time first, and rate-monotonic (RM) are well known algorithms based on deadline, slack time and period respectively. Among this EDF and LLF are dynamic scheduling algorithms as priorities of jobs of periodic task changes dynamically on the basis of deadline and slack time respectively while RM is a static scheduling algorithm as priority of jobs of periodic task is static on the bases of period.

All these algorithms performance differs in overloaded and under loaded condition. Dynamic scheduling algorithms perform optimum in under loaded condition but as system become slightly overloaded their performance deteriorate very badly. Whereas static scheduling algorithms do not perform optimally in under loaded condition but performs fairly well in over loaded condition compared to dynamic scheduling algorithm. None of dynamic or static algorithm is best for both under-loaded as well as overloaded condition.

Our aim is to combine the advantageous features of both dynamic and static real-time scheduling algorithms together. That is performance of dynamic algorithm in under-loaded condition and performance of static algorithm in over-loaded condition. In this paper we enhanced the LLF dynamic real-time scheduling algorithm with RM static real-time scheduling algorithm. In under-loaded condition scheduler schedules jobs according to LLF whereas in overloaded condition it schedules according to RM.

General Terms

Dynamic scheduling of Real time systems, Performance enhancement of scheduling algorithm in both underloaded and overloaded condition of the scheduler.

Keywords

EDF, LLF, RM, Scheduling Algorithms, Real-Time operating Systems.

1. INTRODUCTION

Real-time system is required to complete its work and deliver its services on the basis of time. 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[3]. Therefore, real-time systems have well defined, fixed time constraints i.e. processing must be done within the defined constraints otherwise the system will fail. Real-time systems can be categorized in two basic types: Hard and Soft. In hard real-time systems, all jobs must complete execution prior to their deadline - a missed deadline constitutes a system failure. Such systems are used where the consequences of missing a deadline may be serious or even disastrous. A soft real-time system is less restrictive. Jobs may continue execution beyond their deadlines at some penalty - deadlines are considered as guidelines, and the system tries to minimize the penalties associated with missing them. Such systems are used when the consequences of missing deadlines are smaller than the cost of meeting them in all possible circumstances. Cell phones and multimedia applications would both use soft real-time systems. [2]

1.1 Real-Time Scheduler

Real-time scheduler schedules jobs of various periodic tasks. Every periodic task delivers jobs at particular period. Real-time scheduling techniques can be categorized in two basic types: Static and Dynamic. Static algorithm assigns all priorities at design time, and it remains constant for the lifetime of a task. Dynamic algorithm assigns priority at runtime, based on execution parameters of tasks. Dynamic scheduling can be either with static priority or dynamic priority. RM (Rate Monotonic) and DM (Deadline Monotonic) are examples of dynamic scheduling with static priority [1]. EDF (Earliest Deadline First) and LLF (Least Laxity First) are examples of dynamic scheduling with dynamic priority. Dynamic priority algorithms can be divided into two categories depending on whether individual jobs can change priority while they are active. – In job-level fixed-priority algorithms, jobs cannot change priorities. EDF is a job-level fixed-priority algorithm. – On the other hand, in job-level dynamic-priority algorithms, jobs may change priority during execution. For example, the Least Laxity First (LLF) algorithm [1] is a job-level dynamic-priority algorithm. At time t, the laxity of a job is (d - t - f), where d is the jobs deadline and f is its remaining execution requirement. Here, the laxity is the maximum amount of time a job may be forced to wait if it were to execute on a processor and still meet its deadline. The LLF algorithm assigns higher priority to jobs with smaller laxity. Since the laxity of a job can change over time, the job priorities can change dynamically.

2. THE SCHEDULING ALGORITHMS

2.1 The LLF Scheduling Algorithm

In single processor systems, the least laxity algorithm is another optimal algorithm. At any scheduling decision instant the task with the shortest laxity l, i.e., the difference between the deadline interval d and the computation time c is assigned the highest dynamic priority.

$$d - c = l$$

The Least-Laxity (LL) algorithm is a dynamic preemptive scheduling algorithm based on dynamic task priorities. The task with the shortest laxity is assigned the highest dynamic priority [8].LLF algorithm compares the Slack time of all jobs ready to run and declare the most important job having the smallest Slack time.

Let $c(i)$ denote the remaining computation time of a task at time i .At the arrival time of a task, $c(i)$ is the computation time of this task. Let $d(i)$ denote the deadline of a task relative to the current time i. Then the laxity (or slack) of a task at time i is $d(i) - c(i)$. Thus the laxity of a task is the maximum time the task can delay execution without missing its deadline in the future. The LL scheduler executes at every instant the ready task with the smallest laxity. If more than one task has the same laxity, LL randomly selects one for execution next [9].

2.2 The RM Scheduling Algorithm

The rate monotonic algorithm is a dynamic preemptive algorithm based on static task priorities [8].

RM is a fixed-(static-) priority scheduler using the task’s (fixed) period as the task’s priority. RM executes at any time instant the instance of the ready task with the shortest period first. If two or more tasks have the same period, then RM randomly selects one for execution next [9].

Scheduling under the two algorithms referred are shown in Figure 1 for the first few jobs of a task system with two tasks T1 (1, 3, 5) and T2 (0, 4, 10). The two algorithms differ in the complexity of their priority schemes and their ability to meet the timing constraints and form the basis of a priority-based classification of scheduling algorithms [7].

2.3 The LLF_RM Scheduling Algorithm

As LLF perform optimum in under-loaded condition but not in over-loaded condition we tried to take an advantage of RM algorithm in overloaded condition by enhancing LLF algorithm by RM algorithm. Using this LLF_RM algorithm scheduler switches between LLF and RM according to load. As long as system is under-loaded it follows LLF and as system become slightly overloaded it switches to RM.

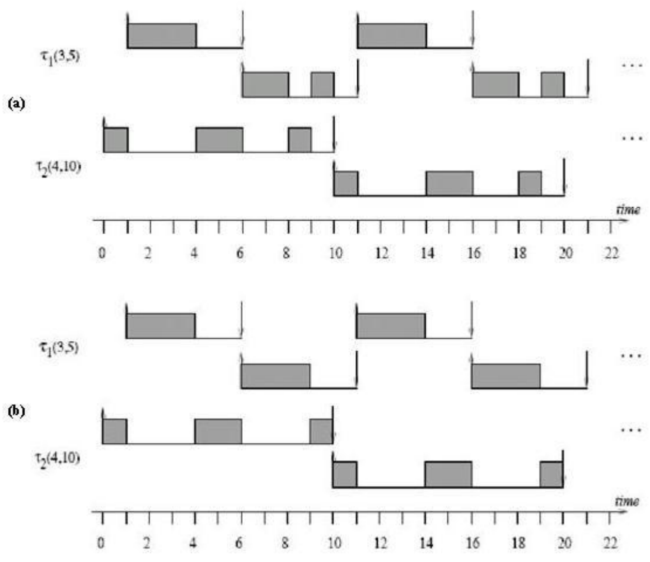

Figure 1: Scheduling Using (a) LLF (b) RM for a single processor system with two tasks T1(3,5) and T2(4,10)

3 SYSTEM AND TASK MODEL

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.

We call each unit of work that is scheduled and executed by the system as a job and a set of related jobs, which jointly provide some system function, is a task [1]. All the tasks are assumed to be periodic.

It has been assumed that the system is not having resource contention problem. Moreover, preemption and the scheduling algorithm incur no overhead. 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. It has been assumed that the systems are having soft timing constraints i.e. soft real-time systems. In soft real-time systems, each task has a positive value. If a task succeeds, then the system acquires its value. If a task fails, then the system gains less value from the task [5].

4. SIMULATION METHOD

We have implemented our algorithms in the same environment and have run simulations to accumulate empirical data. The results of the proposed algorithms are compared with each other in the same environment. LLF is dynamic while RM is a static algorithm. Periodic tasks have been considered for taking the results. For periodic tasks, load of the system can be defined as summation of ratio of executable time and period of each task. We have generated 50 task sets for 16 load values from 0.5 to 3.0. Each task set is having 5 periodic tasks. Each task set is simulated for 500 clock cycles.

The system is said to be overloaded when the tasks offered to the scheduler cannot be feasibly scheduled even by a clairvoyant scheduler 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. Therefore, Success Ratio (SR) and Effective Processor Utilization (EPU) are considered as our main performance measuring criteria and defined as:

1. In real-time systems, deadline meeting is the most important. Therefore, the most appropriate performance metric is the Success Ratio and defined as [6],

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

2. Effective Processor Utilization (EPU) gives information about how efficiently the processor is used and it is defined as [4],

$$\text{EPU} = \sum_{i \in R} \frac{\text{Vi}}{\text{T}}$$

Where,

- V is value of a job and,
  - Value of a job = Execution time of a job, if the job completes within its deadline.
  - Value of a job = 0, if the job fails to meet the deadline.
- R is set of all the jobs which are executed by the CPU.
- T is total time of scheduling.

The results are measured and compared in terms of SR and EPU.

5. FINAL RESULTS

We have taken results for LLF, RM and LLF_RM algorithms for different load conditions. Load of task sets ranges from 0.5 (under loaded) to 3.0 (overloaded). Table 1 shows results in terms of SR and EPU for every algorithm in different load conditions.

Table 1 Load Vs Success Ratio and Load Vs Effective Processor Utilization for LLF, RM and LLF_RM

Load	SR (%)			EPU (%)		
	LLF	RM	LLF_RM	LLF	RM	LLF_RM
0.50	100	100	100	51.98	51.98	51.98
0.60	100	100	100	61.34	61.60	61.34
0.70	100	100	100	70.73	71.15	70.73
0.75	100	100	100	75.93	76.19	75.93
0.80	100	100	100	80.89	81.08	80.89
0.85	100	100	100	85.38	85.60	85.38
0.90	100	99.36	100	90.62	90.40	90.62
0.95	100	98.16	100	95.38	92.89	95.38
1.00	100	89.42	100	99.99	89.05	99.99
1.01	97.48	86.89	97.04	97.55	78.02	95.96
1.02	83.17	86.23	91.24	81.84	74.64	83.78
1.03	66.09	87.35	89.39	66.84	78.52	84.24
1.04	44.24	86.47	87.72	44.37	77.72	81.14
1.50	2.18	79.86	77.82	2.75	72.67	63.89
2.00	0.67	81.13	74.59	0.62	74.87	60.05
3.00	0.13	78.78	73.94	0.06	71.39	57.75

Figure 2 shows comparison of the results of Success Ratio (SR) for LLF, RM and LLF_RM algorithms. The results are taken from under loaded condition (load value=0.50) to overloaded condition (load value=3.00). Results show that LLF is optimal algorithm over single processor system, when system is under loaded, but performance start to degrade exponentially as system goes towards overloaded condition. RM algorithm performance starts to degrade when load increase from 0.85. while enhanced algorithm performs optimal in under-loaded conditions and also performs well in overloaded conditions.

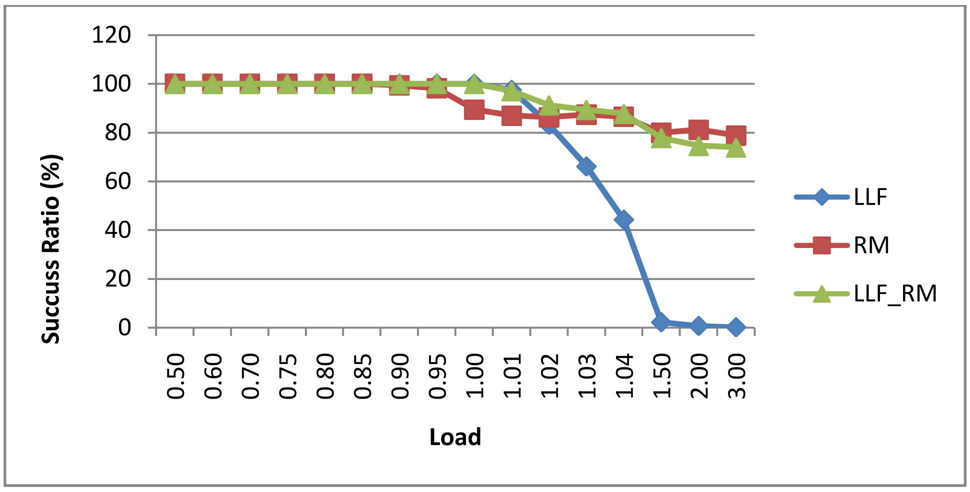

Figure 2 Load Vs Success Ratio for LLF, RM and LLF_RM

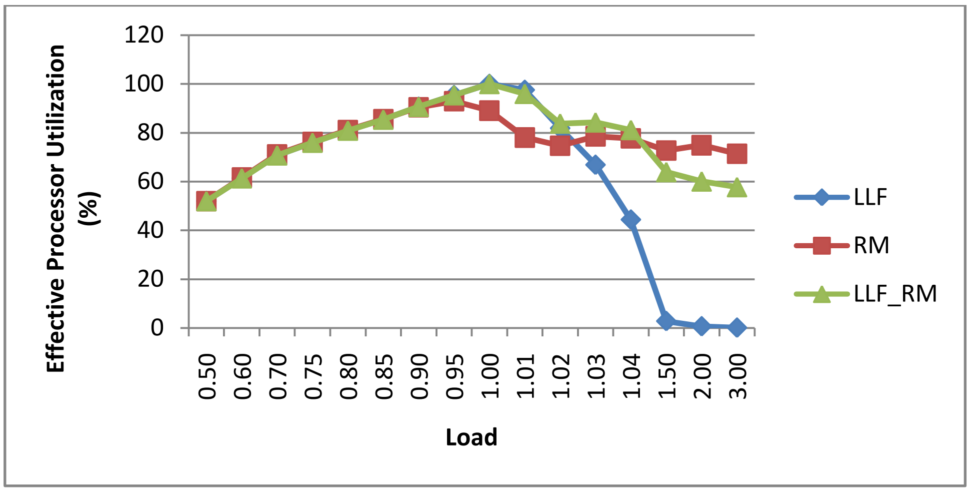

Figure 3 Load Vs Effective Processor Utilization for LLF, RM and LLF_RM

Figure 3 shows comparison of the results of Effective Processor Utilization (EPU) for LLF, RM and LLF_RM algorithms.

6. CONCLUSION

The algorithms discussed in this paper are dynamic scheduling algorithms with dynamic and static priority for real-time single processor systems. LLF and LLF_RM are dynamic scheduling algorithms with dynamic priority, while RM is dynamic scheduling algorithm with static priority.
We can conclude following from the results gained during simulation

- In under loaded condition
  - LLF shows an optimal performance
  - Performance of RM starts to degrade after load 0.85 slightly.
  - LLF_RM shows an optimal performance
- In overloaded condition
  - Performance of LLF starts to degrade exponentially as system become slightly overloaded.
  - RM algorithm performs well.
  - LLF_RM performs well compared to LLF.

7. REFERENCES

[1] Liu C. L., Layland L., Scheduling algorithms for multiprogramming in a hard-realtime environment, Journal of ACM, Vol 20(1), pp. 46-61, 1973.

[2] Ketan Kotecha, Apurva Shah, Adaptive Scheduling Algorithm for real-time operating system, In proceedings of IEEE Congress on Evolutionary Computation (CEC 2008), HongKong, pp. 2109-2112, June 2008.

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

[4] Apurva Shah, Ketan Kotecha, Dipti Shah, Dynamic Scheduling for Real-Time Distributed System using ACO, To Appear in International Journal of Intelligent Computing and Cybernetics, (IJICC).

[5] Locke C. D., Best Effort Decision Making for Real-Time Scheduling, Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University, USA, 1986.

[6] Ramamritham K., Stankovik J. A., Shiah P. F., Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems, IEEE Transaction on Parallel and Distributed Systems, Vol 1(2), pp. 184-194, 1990.

[7] Carpenter J., Funk S. H., Holman P., Srinivasan A., Anderson J., Baruah S., A categorization of real-time multiprocessor scheduling problems and algorithms, In Joseph Y.T Leung, editor, Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press LLC, 2003.

[8] Hermann Kopetz. Real-Time Systems Design Principles for Distributed Embedded Applications Second Edition

[9] ALBERT M. K., REAL-TIME SYSTEMS Scheduling, Analysis, and Verification, CHENG, University of Houston