A Study of Scheduling on Distributed Real-Time Systems

Hisashi Baba (210072)

School of Infomation Science,
Japan Advanced Institute of Science and Technology

2003.2/14

Keywords: Distributed Real-Time, Real-Time Scheduling, Task Allocation, Load Balancing.

1 Introduction

It is important for real-time systems to perform tasks by their deadlines. Various scheduling algorithms have been proposed to meet deadlines.

Recently, rapid technology development enables to establish distributed real-time systems which were difficult to impliment once.

Compared with histories uniprocessor scheduling, most multiprocessor scheduling can not be optimal[3]. In this paper, we propose a task allocation method which respects static priority and laxity on distributed real-time systems.

2 Real-Time Scheduling

Real-time scheduling respects task priority in general. Task priorities are decided in advance by developer, or are based on task properties.

The later type many algorithms have been proposed. As a uniprocessor scheduling, RM (Rate Monotonic), EDF (Earliest Deadline First) etc. are major algorithms. Some algorithms including above two have been proved optimal under certain conditions.

On the other hand, in case of multiprocessor scheduling, it is known that no on-line algorithm is optimal. For off-line scheduling almost problems are NP-complete[3]. In addition, there are inherent anomalies. So heuristics are used in fact.

As uniprocessor shceduling is easier than multiprocessor one, problems are often divided into task allocation and scheduling on each processor.

3 Calculation Load

When the most underloaded processor will be choosed to allocate tasks, it is important to be used which parameter as load descriptor. CPU utilization, number of tasks have been used widely. In this paper, we propose to install the following parameters as load descriptor originated at idea of adaptive dynamic priority scheduling[1].

- Static priority
- Laxity

Processor load ρ is calculated with next equation.

$$\rho_i = \frac{1}{M_i} \sum_{k=1}^{N} (\frac{P_k}{P_{lv}} + \frac{c_k}{X_k})$$

Where M is processor relative speed, P is task static priority, P_{lv} is priority level, c is remain execution time of task, X is laxity of task, N is number of tasks on the processor, respectively. As performing task allocation based on the calculated processor load, we plan to decrease deadline over by load balancing.

4 Evaluation

We asume that a distributed system consists of some processors and a global scheduler which allocate tasks to processors. We implimente a scheduling simulator and evaluate simulation under various conditions. The following are main conditions.

Load Descriptor
Select load descriptor from number of tasks, CPU utilization, CPU load rate, and proposed method.

Scheduling Algorithm
Select scheduling algorithm to choose task by the global scheduler or schedulers on processors from method based on static priority, FCFS, EDF, LLF, RM, DM.

Each Overhead
Scheduling, load calculation, and communication cause overhead. Overhead according to each selection is added. Select simulation with or without overhead.

In addition, select number/speed of processors.

There are vaious cost functions to evaluate scheduling algorithm, in this paper, we use number of late tasks, average response time, and average static priority of late tasks.

5 Conclusion

In this paper, we proposed a new load descriptor for task allocation whose target is on-line scheduling on distributed real-time systems. We installed the idea of adaptive dynamic priority scheduling and laxity of task as load descriptor, considered difference of processor speed.

We implemented a simulator and evaluated simulation result. In case that communication time is short enough, our meghod decrease late tasks whose priority are high.

As future works, we are going to simulate under other conditions, evaluate with other cost functions.

References

[1] Kazumichi Kuritani, "Dynamic Scheduling Using the Priority for the Real-Time OS," Thesis JAIST, School of Infomation Science, Mar. 2003.

[2] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of the ACM, Vol. 20, No. 1, pp. 46–61, 1973.

[3] J. A. Stankovic, M. Spuri, M. D. Natale, and G. C. Buttazzo, "Inplications of Classical Scheduling Results for Real-Time Systems," IEEE Computer, Vol. 28, No. 6, pp. 16–25, June 1995.