A Brief Survey of Multi-Processor Scheduling For Hard Real-Time Systems

Xin Lin[a], Xiaorong Zhu[a], Lijia Liu[a]

[a]*Department of Computer Science, The University of Texas at Austin*

Abstract

In class, both of scheduling algorithms [1] and priority inheritance protocols [2] in the context of a single processor were examined in details. Nevertheless, the emergence and popularity of distributed computing system gave rise to the need to solve multi-processor scheduling and priority inheritance problems. As the supplementary study, this paper surveys existing scheduling algorithms in the context of multiple processors. The very first section outlines the background of multi-processor scheduling problems, as well as system models, terminology, and the metrics of scheduling algorithms. After that, partitioned scheduling and global scheduling, as the primary objects of our research, will be fully explored. Moreover, we will also give brief sketch to the hybrid approaches of partitioned scheduling and global scheduling.

Keywords: System, Scheduling Algorithm, Task Management

1. Introduction

1.1. Problem Defintion

1.2. Preview Of Related works

1.3. Paper Organization

0. Background and introduction
1. System Models
2. Partitioned Scheduling
3. Global Scheduling
4. Hybrid Approach
5. Conclusion and Discussion

2. System Models

2.1.

3. Partitioned Scheduling

In this section, we will review some partitioned approaches to multiprocessor real-time scheduling.

3.1. Characteristic of Partitioned Scheduling

3.2. RMNF

3.3. RMFF

3.4. EDF-FF

3.5. EDF-BF

3.6. Comparision

4. Global Scheduling

In this section, we will dive into another branch of scheduling strategies – global approaches to multiprocessor real-time scheduling.

4.1. Overview

Global scheduling algorithms, as its name suggests, globally schedules any feasible periodic task set. In contrast to partitioned scheduling, global scheduling schedules jobs and tasks in one single shared queue instead of multiple local, dedicated queues. By this means, the automatic load balancing and lower average response time can be achieved by global approaches. In addition to this, global approaches also take advantages of the simpler implementations and the existence of optimal schedulers.

Global scheduling startegies include Global Fixed-Job-Priority Scheduling, Global Fixed-Task-Priority Scheduling, and Global Dynamic Priority Scheduling. Although there are various categories of global scheduling algorithms, the focus of this paper is on the Global Dynamic Priority Scheduling. In the following subsection, it will be chacterized in details.

4.2. Global Dynamic Priority Scheduling

In this subsection, we will present our in-depth exploration to the track of global dynamic priority scheduling algorithm. To the best of our knowledge, a number of global dynamic priority scheduling algorithms are optimal for periodic tasksets with explicit or implicit deadlines. For example, Proportionate Fairness algorithm and its variants including PD, PD^2, ERFair, BF, SA [3], and LLREF [4] as well, are all optimal for offline environment. Nevertheless, no algorithms until now are optimal to cope with online preemptive scheduling problem, where tasksets are sporadic and multi-processor environments are enforced. On the other hand, despite of its optimality and dominance in theory, the usage of global dynamic priority algorithms are limited in practice. This is because the existence of frequent preemption and migration between tasks gives rise to excessive overheads in potential.

The following part of this subsection will provide brief summary of three classic global dynamic priority scheduling algorithms. They are respectively Proportionate Fairness Algorithm (PFair), and Largest Local Remaining Execution First (LLREF).

4.2.1. PFair

Baruah et al [5] introduced Proportionate Fairness Algorithm. The Pfair class of algorithms that allow full migration and fully dynamic priorities have been shown to be theoretically optimal – i.e., they achieve a schedulable utilization bound (below which all tasks meet their deadlines) that equals the total capacity of all processors. Here are some fundamental principles of Proportionate Fairness algorithms:

1. Timeline is divided into equal length slots.
2. Task period and execution time are multiples of the slot size.
3. Each task receives amount of slots proportional to its task utilization.

The essential part of PFair algorithms is the quantum-based optimization defined over the lag of each task $lag(\tau_i, t)$, with the goal of minimizing the maximum lags of all tasks $\max_t |lag(\tau_i, t)|$.

$$\underbrace{lag(\tau_i, t)}_{error} = \underbrace{t \cdot (\frac{c_i}{T_i})}_{fluid\ exectuion\ in[0,t)} - \underbrace{allocated(\tau_i, t)}_{real\ execution\ in[0,t)} \tag{1}$$

The generation of an optimal schedule is based on the above definition of lag. PFair algorithm does execute all urgent tasks with $lag(\tau_i, t) > 0$ and $lag(\tau_i, t + 1) \geq 0$ if τ_i executes. On top of that, PFair algorithm does not execute tnegru tasks, for which $lag(\tau_i, t) < 0$ and $lag(\tau_i, t + 1) \leq 0$ if τ_i does not execute. Besides, for other tasks, only those that have the least t such that $lag(\tau_i, t) > 0$ are executed.

The PFair algorithm will assign priorities to tasks at every time slot, which indicated itself as one of job-level dynamic priority scheduling policies. However, this characteristic gives rise to some issues, for example, frequent preemptions and frequent migrations.

Proportionate Fairness algorithm, as a strong candidate for solving resource allocation problems, has wide variety of interesting applications and powerful theoretical support. For instances, Kelly et al [6] presented its application on the problem of rate control for communication networks. Application of PFair algorithm on LANs and hoc networks was fulfilled by Jiang et al [7]. Besides, Bonald's paper [8] provides in-depth queueing analysis over proportionate fairness as well as max-min fairness and balanced fairness.

4.2.2. LLREF

LLREF was firstly introduced by Cho et al [4]. Similar to PFair class of algorithms, LLREF is also based on the fluid scheduling model, where each task executes at a constant rate at all times. The principal idea of LLREF is that given M processors, M largest local remaining execution time tasks are selected first for every secondary event. This is also called the LLREF scheduling policy. To reason over task execution behavior on multiprocessors, a novel abstraction called Time and Local Execution Time Domain Plane (T-L Plane) was developed.

This algorithm divides the schedule into Time and Local execution time planes (TL-planes), which are determined by task deadlines. The algorithm schedules tasks by creating smaller local jobs within each TL-plane. The only parameters considered by the algorithm during a TL-plane are the parameters of the local jobs. When a TL-plane completes, the next TL-plane is started. The duration of each TL-plane is the amount of time between consecutive deadlines.

4.3. Summary

The global scheduling paradigm has advantages over the partitioned approach. First of all, if tasks can join and leave the system at run-time, then it may be necessary to reallocate tasks to processors in the partitioned approach. In addition, the partitioned approach cannot produce optimal real-time schedules – one that meets all task deadlines when task utilization demand does not exceed the total processor capacity – for periodic task sets, since the partitioning problem is analogous to the bin-packing problem which is known to be NP-hard in the strong sense. On top of that, in some embedded processor architectures with no cache and simpler structures, the overhead of migration has a lower impact on the performance. Finally, global scheduling can theoretically contribute to an increased understanding of the properties and behaviors of real-time scheduling algorithms for multiprocessors.

However, the global scheduling paradigm has also several disadvantages. Firstly, global scheduling strategies are much more complicated to implement than partitioned scheduling. In other words, for the partitioned approach, once a set of tasks are allocated to processors, the multiprocessor real-time scheduling problem becomes a collection of single processor real-time scheduling problems. The ease of programming partitioned scheduling is obvious since the single processor scheduling problem has already been well-studied and optimal algorithms with easy implementations already exist. Secondly, migrating tasks at run-time means more runtime overhead in that migrating tasks may suffer cache misses on the newly assigned processor. If the task set is fixed and known in advanced, it is obvious that the partitioned approach provides more appropriate solutions.

5. Hybrid Approaches

6. Conclusions

References

[1] C. L. Liu, J. W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the ACM (JACM) 20 (1973) 46–61.

[2] L. Sha, R. Rajkumar, J. P. Lehoczky, Priority inheritance protocols: An approach to real-time synchronization, Computers, IEEE Transactions on 39 (1990) 1175–1185.

[3] A. Khemka, R. Shyamasundar, An optimal multiprocessor real-time scheduling algorithm, Journal of parallel and distributed computing 43 (1997) 37–45.

[4] H. Cho, B. Ravindran, E. D. Jensen, An optimal real-time scheduling algorithm for multiprocessors, in: Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International, IEEE, pp. 101–110.

[5] S. K. Baruah, N. K. Cohen, C. G. Plaxton, D. A. Varvel, Proportionate progress: A notion of fairness in resource allocation, Algorithmica 15 (1996) 600–625.

[6] F. P. Kelly, A. K. Maulloo, D. K. Tan, Rate control for communication networks: shadow prices, proportional fairness and stability, Journal of the Operational Research society (1998) 237–252.

[7] L. B. Jiang, S. C. Liew, Proportional fairness in wireless lans and ad hoc networks, in: Wireless Communications and Networking Conference, 2005 IEEE, volume 3, IEEE, pp. 1551–1556.

[8] T. Bonald, L. Massoulié, A. Proutiere, J. Virtamo, A queueing analysis of max-min fairness, proportional fairness and balanced fairness, Queueing systems 53 (2006) 65–84.