

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

11 **2. System Models**

12 *2.1.*

13 **3. Partitioned Scheduling**

14 In this section, we will review some partitioned approaches to multipro-
15 cessor real-time scheduling.

16 *3.1. Characteristic of Partitioned Scheduling*

17 *3.2. RMNF*

18 *3.3. RMFF*

19 *3.4. EDF-FF*

20 *3.5. EDF-BF*

21 *3.6. Comparision*

22 4. Global Scheduling

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

25 4.1. Overview

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

33 Global scheduling strategies include Global Fixed-Job-Priority Scheduling,
34 Global Fixed-Task-Priority Scheduling, and Global Dynamic Priority
35 Scheduling. Although there are various categories of global scheduling algorithms,
36 the focus of this paper is on the Global Dynamic Priority Scheduling.
37 In the following subsection, it will be characterized in details.

38 4.2. Global Dynamic Priority Scheduling

39 In this subsection, we will present our in-depth exploration to the track
40 of global dynamic priority scheduling algorithm. To the best of our knowledge,
41 a number of global dynamic priority scheduling algorithms are optimal
42 for periodic tasksets with explicit or implicit deadlines. For example, Proportionate
43 Fairness algorithm and its variants including PD, PD², ERFair, BF, SA [3],
44 and LLREF [4] as well, are all optimal for offline environment.
45 Nevertheless, no algorithms until now are optimal to cope with online preemptive
46 scheduling problem, where tasksets are sporadic and multi-processor environments
47 are enforced. On the other hand, despite of its optimality and dominance in theory,
48 the usage of global dynamic priority algorithms are limited in practice. This is
49 because the existence of frequent preemption and migration between tasks gives
50 rise to excessive overheads in potential.

51 The following part of this subsection will provide brief summary of three
52 classic global dynamic priority scheduling algorithms. They are respectively
53 Proportionate Fairness Algorithm (PFair), and Largest Local Remaining Execution
54 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.

4.2.2. *LLREF*

LLREF was firstly introduced by Cho et al [4]. LLREF was designed based on a novel abstraction for reasoning over task execution behavior on multiprocessors, called Time and Local Execution Time Domain Plane (T-L Plane).

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

⁹⁰ is fixed and known in advanced, it is obvious that the partitioned approach
⁹¹ provides more appropriate solutions.

⁹² **5. Hybrid Approaches**

⁹³ **6. Conclusions**

94 **References**

- 95 [1] C. L. Liu, J. W. Layland, Scheduling algorithms for multiprogramming
96 in a hard-real-time environment, *Journal of the ACM (JACM)* 20 (1973)
97 46–61.
- 98 [2] L. Sha, R. Rajkumar, J. P. Lehoczky, Priority inheritance protocols: An
99 approach to real-time synchronization, *Computers, IEEE Transactions*
100 on 39 (1990) 1175–1185.
- 101 [3] A. Khemka, R. Shyamasundar, An optimal multiprocessor real-time
102 scheduling algorithm, *Journal of parallel and distributed computing* 43
103 (1997) 37–45.
- 104 [4] H. Cho, B. Ravindran, E. D. Jensen, An optimal real-time scheduling
105 algorithm for multiprocessors, in: *Real-Time Systems Symposium, 2006.*
106 *RTSS’06. 27th IEEE International, IEEE*, pp. 101–110.
- 107 [5] S. K. Baruah, N. K. Cohen, C. G. Plaxton, D. A. Varvel, Proportionate
108 progress: A notion of fairness in resource allocation, *Algorithmica* 15
109 (1996) 600–625.