Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

An optimal boundary fair scheduling algorithm for multiprocessor real-time systems

Dakai Zhu [a,*], Xuan Qi [a], Daniel Mossé [b], Rami Melhem [b]

[a] University of Texas at San Antonio, San Antonio, TX 78249, United States
[b] University of Pittsburgh, Pittsburgh, PA 15260, United States

ARTICLE INFO

Article history:
Received 6 September 2010
Received in revised form
12 May 2011
Accepted 16 June 2011
Available online 26 June 2011

Keywords:
Real-time systems
Multiprocessor
Periodic tasks
Optimal scheduling algorithms
Boundary fairness (Bfair) scheduling

ABSTRACT

With the emergence of multicore processors, the research on multiprocessor real-time scheduling has caught more researchers' attention recently. Although the topic has been studied for decades, it is still an evolving research field with many open problems. In this work, focusing on periodic real-time tasks with *quantum-based* computation requirements and implicit deadlines, we propose a novel optimal scheduling algorithm, namely *boundary fair (Bfair)*, which can achieve full system utilization as the well-known Pfair scheduling algorithms. However, different from Pfair algorithms that make scheduling decisions and enforce proportional progress (i.e., *fairness*) for all tasks at *each and every* time unit, Bfair makes scheduling decisions and enforces fairness to tasks *only* at tasks' period boundaries (i.e., deadlines of periodic tasks). The correctness of the Bfair algorithm to meet the deadlines of all tasks' instances is formally proved and its performance is evaluated through extensive simulations. The results show that, compared to that of Pfair algorithms, Bfair can significantly reduce the number of scheduling points (by up to 94%) and the overhead of Bfair at each scheduling point is comparable to that of the most efficient Pfair algorithm (i.e., PD^2). Moreover, by aggregating the time allocation of tasks for the time interval between consecutive period boundaries, the resulting Bfair schedule can dramatically reduce the number of required context switches and task migrations (as much as 82% and 85%, respectively) when compared to those of Pfair schedules, which in turn reduces the run-time overhead of the system.

1. Introduction

For different (e.g., periodic, sporadic and aperiodic) real-time tasks to be executed on systems with a single or multiple processing units, the scheduling problem of how to guarantee various hard and/or soft timing constraints has been studied extensively in the last few decades [42]. Although the scheduling theory for uniprocessor real-time systems has been well developed, such as the optimal EDF (earliest deadline first) and RM (rate-monotonic) scheduling algorithms [35], the scheduling for multiprocessor real-time systems is still an evolving research field and many problems remain open due to their intrinsic difficulties. With the emergence of multicore processors, there is a reviving interest in scheduling algorithms for multicore/multiprocessor real-time systems and many interesting results have been reported in recent years, such as [5,9–11,20,24,27,26,31,33,34].

Traditionally, there are two major approaches for scheduling real-time tasks in multiprocessor systems: *partitioned* and *global* scheduling [19,21]. In partitioned scheduling, each task is assigned to a specific processor and processors can only execute the tasks that are assigned to them. Although the well-established uniprocessor scheduling algorithms (such as EDF and RMS [35]) can be employed on each processor after partitioning tasks to processors, finding a feasible partition of tasks to processors has been shown to be NP-hard [19,21]. For the global scheduling, on the other hand, all ready tasks are put into a shared single queue and each idle processor fetches the next highest priority ready task from the global queue for execution. Despite its flexibility that allows tasks to migrate and execute on different processors, it has been shown that simple global scheduling policies (e.g., global-EDF and global-RMS) could fail to schedule task sets with extremely low system utilization [21]. In addition, neither partitioned nor global scheduling dominates one another as there are task sets that can be scheduled by one approach but not the other, and vice versa.

Recently, as a hierarchical approach, *cluster scheduling* has been investigated. In this approach, processors are grouped into clusters and tasks are partitioned among different clusters. For tasks that are allocated to a cluster, different global scheduling policies (e.g., global-EDF) can be adopted within the cluster [9,43]. Note that, cluster scheduling is a *general* approach, which will reduce to partitioned scheduling when there is only one processor in each cluster. For the case of a single cluster containing all the processors, it will reduce to the global scheduling.

* Corresponding author.
E-mail address: dzhu@cs.utsa.edu (D. Zhu).

0743-7315/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2011.06.003

Since the shared cache architecture in multicore processors can significantly alleviate the task migration overhead in the global scheduling, in this work, we study an *optimal* global scheduling algorithm for a set of periodic real-time tasks with implicit deadlines (i.e., the relative deadlines of tasks equal their periods), which can achieve full system utilization. Fairness has been traditionally utilized to guarantee quality of service (QoS) in computing systems (e.g., in wireless networks [29]). In [12], Baruah et al. exploited fairness as a vehicle to design the first well-known quantum-time-based optimal global scheduling algorithm. Specifically, the *proportional fair (Pfair)* scheduler enforces proportional progress (i.e., *fairness*) for all tasks at *each and every* time unit, which can achieve full system utilization while guaranteeing that all tasks meet their deadlines. Several sophisticated Pfair variations have also been studied, such as *PD* [13] and *PD*^{2} [3]. Recently, assuming that the time domain is *continuous*, the *T–L Plane* based algorithms were studied, which can also achieve full system utilization [15,23]. Following this line of research, a generalized deadline-partitioned fair (DP-Fair) scheduling model was investigated in [34]. However, since each task needs to get its time share within any allocation interval (e.g., time interval between adjacent deadlines of tasks), the time allocation to tasks can be arbitrarily small, which can lead to extremely high scheduling overhead for those scheduling algorithms.

Note that, the proportional fairness is actually a much more strict requirement than that of the original scheduling problem, where the only requirement is to have each task instance get enough time allocation and complete its execution before its deadline. Moreover, by making scheduling decisions at *each and every* quantum time unit, the Pfair algorithms can also lead to high scheduling overhead. Observing the fact that a periodic real-time task can only miss its deadline at its period boundary, in this work, we develop an optimal *boundary fair (Bfair)* scheduling algorithm, which makes scheduling decisions and ensures fairness for tasks *only* at their period boundaries (which are tasks' arrival times as well). Specifically, at each period boundary, the Bfair algorithm will allocate processors to tasks for the time units between the current period boundary and the next period boundary of tasks. Similar to the Pfair algorithms, to prevent deadline misses, Bfair also ensures *fairness* for tasks, but only at the period boundaries. That is, at each period boundary time, the allocation error for any task is less than one time unit.

We have formally proved the correctness of the Bfair algorithm on meeting the deadlines of all tasks' instances while achieving 100% system utilization. The time complexity for an efficient implementation of Bfair can be $O(n)$ (where n is the number of tasks in the system) at each scheduling point, which is the same as that of the Pfair algorithm [12]. However, Bfair can significantly reduce the number of scheduling points (by up to 94% in our simulations), and thus reduce the overall scheduling overhead. Moreover, our simulations show that the time overhead of Bfair for each scheduling point as well as for generating the whole schedule is comparable to that of the most efficient Pfair algorithm, *PD*^{2} [3]. Furthermore, by aggregating the time allocation of tasks for the time interval between consecutive period boundaries, the resulting Bfair schedule can also dramatically reduce the number of context switches and task migrations, as much as 82% and 85%, respectively, when compared to those of Pfair schedules. Such reduction in context switches and task migrations can significantly reduce the run-time overhead, which is specially valuable for real-time systems.

There are several contributions of this work:

- First, we introduce the concept of *boundary fairness* for the periodic real-time scheduling problem, which is *fair enough* to get a feasible schedule while only requires scheduling decisions at tasks' period boundaries (i.e., deadlines and arrival times of tasks);
- Second, we propose an *optimal* and *efficient Bfair scheduling* algorithm and prove its correctness to generate a feasible schedule while achieving full system utilization;
- Finally, we evaluate the proposed Bfair algorithm and show its superior performance on reducing scheduling overhead when comparing to Pfair algorithms through extensive simulations.

The remainder of this paper is organized as follows. The related work is reviewed in Section 2. Section 3 defines the related notations and formulates the problem, which is further illustrated through a concrete motivating example. Section 4 presents the Bfair algorithm and its complexity analysis. The correctness of the Bfair algorithm is formally proved in Section 5. Simulation results are reported and discussed in Section 6. Section 7 gives out our conclusions.

2. Related work

Although rate-monotonic (RM) scheduling and earliest deadline first (EDF) have been shown to be optimal in uniprocessor periodic real-time systems, for static and dynamic priority assignments, respectively [35], neither of them is optimal for multiprocessor real-time systems [21]. Based on the partitioned scheduling, Oh and Baker studied the rate-monotonic first-fit (RMFF) heuristic and showed that RMFF can schedule any system of periodic tasks with total utilization bounded by $m(2^{1/2} - 1)$, where m is the number of processors in the system [41]. Later, a better bound of $(m + 1)(2^{1/(m+1)} - 1)$ for RMFF was shown in [37]. In [6], Andersson and Jonsson proved that the system utilization bound can reach 50% for a partitioned RM scheduling by exploiting the harmonicity of tasks' periods. For the partitioned scheduling with earliest deadline first (EDF) first-fit heuristic, Lopez et al. showed that any task set can be successfully scheduled if the total utilization is no more than $(\beta \cdot m + 1)/(\beta + 1)$, where $\beta = \lfloor 1/\alpha \rfloor$ and α is the maximum task utilization of the tasks considered [38]. Following similar techniques, the utilization bounds for partitioned-EDF in *uniform* multiprocessor systems (where the processors have the same functionalities but different processing speeds) were developed by Darera in [17].

For global scheduling based EDF, it has been shown that a task set is schedulable on m processors if the total utilization does not exceed $m(1 - \alpha) + \alpha$ [25]. Similarly, for global-RMS scheduling, system utilization of $(m/2)(1 - \alpha) + \alpha$ can be guaranteed [8]. Andersson et al. also studied one scheduling algorithm named RM-US, where tasks with utilization higher than some threshold θ have the highest priority [4]. For $\theta = \frac{1}{3}$, Baker showed that RM-US can guarantee a system utilization of $(m + 1)/3$ [8].

To further improve the achievable system utilization and reduce the scheduling overhead, Andersson et al. proposed the EKG algorithm based on the concept of *portion tasks* [7]. In EKG, a separator is defined as $SEP = \frac{k}{k+1}$ for cases where $k < m$ (m is the number of processors) and $SEP = 1$ for the case of $k = m$. For *heavy* tasks with utilization being more than *SEP*, they are allocated to their dedicated processors following the conventional partitioned-EDF scheduling. Each *light* task (with utilization being no more than *SEP*) is split into two portion tasks only if necessary, where the portion tasks are assigned to adjacent processors. The worst-case system utilization bound that can be achieved by EKG is 66% when it is configured with $k = 2$ that has few preemptions. Full (i.e., 100%) system utilization can be achieved by setting $k = m$, which can result in high scheduling overhead with many preemptions as EKG may allocate arbitrarily small share of time to tasks. Following the same line of research, several semi-partitioned based scheduling algorithms have been proposed very recently, which are different in how to handling portion tasks and thus achieve different system utilizations [26,30–32].