Journal of Systems Architecture 69 (2016) 1–14

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Preference-oriented fixed-priority scheduling for periodic real-time tasks

Rehana Begam[a], Qin Xia[b,*], Dakai Zhu[a,1], Hakan Aydin[c,2]

[a] Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[b] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, 710049, China
[c] Department of Computer Science, George Mason University, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 6 May 2016
Revised 20 July 2016
Accepted 30 July 2016
Available online 30 July 2016

Keywords:
Real-time systems
Periodic tasks
Fixed-priority scheduling
Preference-oriented executions

ABSTRACT

Traditionally, real-time scheduling algorithms prioritize tasks solely based on their timing parameters and cannot effectively handle tasks that have different *execution preferences*. In this paper, for a set of periodic real-time tasks running on a single processor, where some tasks are preferably executed *as soon as possible (ASAP)* and others *as late as possible (ALAP)*, we investigate *Preference-Oriented Fixed-Priority (POFP)* scheduling techniques. First, based on Audsley's *Optimal Priority Assignment (OPA)*, we study a *Preference Priority Assignment (PPA)* scheme that attempts to assign ALAP (ASAP) tasks lower (higher) priorities, whenever possible. Then, by considering the *non-work-conserving* strategy, we exploit the *promotion times* of ALAP tasks and devise an online dual-queue based POFP scheduling algorithm. Basically, with the objective of fulfilling the execution preferences of all tasks, the POFP scheduler retains ALAP tasks in the *delay queue* until their promotion times while putting ASAP tasks into the *ready queue* right after their arrivals. In addition, to further expedite (delay) the executions of ASAP (ALAP) tasks using system slack, runtime techniques based on dummy and wrapper tasks are investigated. The proposed schemes are evaluated through extensive simulations. The results show that, compared to the classical fixed-priority *Rate Monotonic Scheduling (RMS)* algorithm, the proposed priority assignment scheme and POFP scheduler can achieve significant improvement in terms of fulfilling the execution preferences of both ASAP and ALAP tasks, which can be further enhanced at runtime with the wrapper-task based slack management technique.

1. Introduction

Real-time systems, where application tasks generally have deadlines, have been studied for decades. For various task and system models, many classical real-time scheduling algorithms have been proposed by the research community. For example, for periodic real-time tasks running on a single processor system, *Rate Monotonic Scheduling (RMS)* and *Earliest Deadline First (EDF)* are the optimal schedulers for static and dynamic priority based preemptive scheduling algorithms, respectively [15]). With the exclusive focus on meeting tasks' timing constraints, most existing real-time scheduling algorithms prioritize and schedule tasks solely based on their timing parameters (e.g., deadlines and periods). Moreover, to complete the executions of tasks at their earliest possible time instants, these algorithms normally adopt the *work-conserving* strategy, which does not leave the processor idle when there are ready tasks for executions.

However, there are occasions where it can be beneficial to execute tasks at their latest times. For instance, when both *periodic* hard real-time tasks and *aperiodic* soft real-time tasks share the same computing platform, the executions of periodic hard real-time tasks can be maximally delayed in order to get better response times for aperiodic soft real-time tasks [6,7]. In addition, in fault-tolerant systems that adopt the *primary/backup* model, backup tasks should also be executed as late as possible to reduce the overlapped executions with their corresponding primary tasks (which are executed early on other processors) and thus system overheads to achieve cost efficient fault tolerance [12,17].

In contrast to the EDF and RMS schedulers [15] that execute tasks at their earliest possible times, the *Earliest Deadline Latest (EDL)* [6] and *Dual-Priority (DP)* schedulers [7] have been proposed to schedule periodic real-time tasks at their latest times. However, all these classical real-time schedulers treat **all** periodic tasks *uniformly* when assigning priorities to tasks and making scheduling

* Corresponding author.
E-mail addresses: qin.xia@mail.xjtu.edu.cn, xq_computer@sina.com (Q. Xia), dakai.zhu@utsa.edu (D. Zhu), aydin@cs.gmu.edu (H. Aydin).
[1] Member, IEEE
[2] Member, IEEE

http://dx.doi.org/10.1016/j.sysarc.2016.07.005
1383-7621/

decisions. That is, these schedulers do *not* differentiate the execution preferences of tasks and thus **cannot** effectively handle co-running tasks that have different execution preferences.

For a set of periodic real-time tasks running on a single processor that have different execution preferences, where some tasks are preferably executed *as soon as possible (ASAP)* while others *as late as possible (ALAP)*, we have recently studied an online *Preference-Oriented Earliest Deadline (POED)* scheduling algorithm [10]. Basically, different from the EDF/EDL schedulers that consider only the deadlines of tasks when making scheduling decisions, the POED scheduler takes the execution preferences of tasks into consideration and puts ASAP tasks in the central stage. That is, whenever there are ready ASAP tasks at a scheduling point, these tasks can be picked for executions even if they have later deadlines, provided that such executions do not cause deadline misses for other tasks in the future. The POED scheduler has been exploited in fault-tolerant systems to reduce run-time overhead and improve energy efficiency [11]. In addition, such schedulers can also be utilized in mixed-criticality systems to differentiate the executions of high-criticality and low-criticality tasks to better explore system slack at runtime [16].

Although POED was the first scheduler that addresses the different execution preferences of periodic tasks, it is a dynamic priority based scheduler. To the best of our knowledge, no fixed-priority based preference-oriented scheduling algorithm has been reported yet in the literature. Our preliminary study on the *Preference-Oriented Fixed-Priority (POFP)* scheduler was presented in [4], which is extended in this paper to address the topic more thoroughly. First, based on Audsley's *Optimal Priority Assignment (OPA)* algorithm [3], we study a *Preference Priority Assignment (PPA)* scheme for a set of periodic real-time tasks with ASAP or ALAP execution preferences to be executed on a single processor system under preemptive fixed-priority scheduling. With the intuition that low priority tasks are executed late at runtime, the basic idea of PPA is to favor ALAP tasks when assigning lower priorities (and thus ASAP tasks can implicitly have higher priorities) without sacrificing system schedulability. By taking tasks' execution preferences into consideration, the resulting PPA priorities can most likely delay the executions of ALAP tasks while executing ASAP tasks at earlier times.

Then, observing that there are normally idle intervals in fixed-priority schedules, we propose an online preemptive POFP scheduler that adopts the *non-work-conserving* scheduling strategy. Here, to exploit idle intervals and systematically delay (expedite) the executions of ALAP (ASAP) tasks, POFP utilizes a *dual-queue* based scheduling approach. ASAP tasks enter the *ready queue* immediately after they arrive. In contrast, an ALAP task is put into a *delay queue* at its arrival time and will wait there until its *promotion time*, which can be derived following the ideas in the dual-priority scheduling framework [7]. After that, the ALAP task is *promoted* to the ready queue. Such a dual-queue scheduling approach can effectively prevent an ALAP task from being executed before its promotion time, which can in turn give low priority ASAP tasks the opportunity to run at earlier times.

Moreover, we investigate runtime techniques to further delay (expedite) the executions of ALAP (ASAP) tasks by exploiting system slack, which can be expected at runtime as real-time tasks typically take a small fraction of their worst-case execution times [9]. In summary, the contributions of this work are as follows:

- A preference priority assignment (PPA) scheme that explicitly incorporates the ASAP and ALAP execution preferences of periodic real-time tasks is proposed;
- An online dual-queue based preference-oriented fixed-priority (POFP) scheduler is proposed [4], which is preemptive and non-work-conserving in nature to better serve the tasks' ASAP and ALAP execution preferences;
- Runtime techniques are also investigated to further delay (expedite) the executions of ALAP (ASAP) tasks by exploiting system slack at runtime;
- The proposed PPA scheme, POFP scheduler and runtime techniques are evaluated through extensive simulations with synthetic tasks, which are shown to be very effective to fulfill the tasks' ASAP and ALAP execution requirements, when compared to that of the *preference-oblivious* RMS scheduler.

The rest of this paper is organized as follows. We review the closely related work in Section 2. Section 3 presents system models and necessary preliminaries. The preference-oriented priority assignment (PPA) scheme is discussed in Section 4. In Section 5, the preference-oriented fixed-priority (POFP) scheduler is proposed. The runtime techniques are further addressed in Section 6. Section 7 discusses the evaluation results and Section 8 concludes the paper.

2. Closely related work

In the past few decades, real-time systems have been studied extensively and numerous scheduling algorithms have been proposed for different task and system models. In this section, for periodic real-time tasks running on a single processor system, we review closely related scheduling algorithms.

The *Earliest-Deadline-First (EDF)* scheduling algorithm has been the well-known optimal scheduler based on the dynamic priority approach, where the instances (or jobs) of the same task may have different priorities [15]. Specifically, EDF prioritizes task instances (jobs) based on their absolute deadlines, where jobs with smaller deadlines have higher priorities (for jobs that have the same deadlines, the order of their priorities can be arbitrarily assigned without affecting tasks' schedulability). It has been shown that EDF can successfully schedule a set of periodic tasks on a single processor as long as the system utilization is no more than 1.0 [15].

In comparison, in the fixed-priority scheduling, priorities are assigned to tasks and the instances (jobs) of a task have the same priority of that task. For instance, as a classical optimal fixed-priority scheduler, the *Rate Monotonic Scheduling (RMS)* algorithm prioritizes tasks according to their periods, where tasks with smaller periods have higher priorities [15]. That is, if a set of periodic tasks can be feasibly scheduled with fixed-priority scheduling on a single processor, the task set can be successfully scheduled under RMS. Instead of prioritizing tasks solely based on their periods, in [3], Audsley studied an *Optimal Priority Assignment (OPA)* algorithm for periodic tasks under fixed-priority scheduling based on the concept of response time for tasks [1,13]. A comprehensive review on various priority assignment schemes for fixed-priority scheduling can be found in a recent report [8].

The well-known *work-conserving* schedulers EDF and RMS do not leave the processor idle if there are ready tasks for execution, where tasks are executed at their *earliest* times. However, for mixed workload with both hard real-time periodic tasks and soft real-time aperiodic tasks sharing a computing platform, it would be necessary for periodic tasks running at their *latest* times to provide better response times for soft aperiodic tasks. For such a purpose, the *earliest deadline latest (EDL)* algorithm has been developed [6]. By considering all instances of periodic tasks within the least common multiple (LCM) of their periods, EDL generates an offline static schedule to find out their latest execution times while ensuring that there is no deadline miss.

With the same objective, the *dual-priority (DP)* scheme has been developed for fixed-priority scheduling in order to improve the responsiveness of soft real-time aperiodic tasks [7]. The DP scheduler