Journal of Systems Architecture 61 (2015) 127–139

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Preference-oriented real-time scheduling and its application in fault-tolerant systems

Yifeng Guo [a], Hang Su [a], Dakai Zhu [a,*], Hakan Aydin [b]

[a] Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[b] Department of Computer Science, George Mason University, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 26 August 2014
Received in revised form 14 November 2014
Accepted 21 December 2014
Available online 3 January 2015

Keywords:
Periodic real-time tasks
Preference-oriented execution
Scheduling algorithms
Fault-tolerant systems

ABSTRACT

In this paper, we consider a set of real-time periodic tasks where some tasks are preferably executed *as soon as possible (ASAP)* and others *as late as possible (ALAP)* while still meeting their deadlines. After introducing the idea of *preference-oriented (PO) execution*, we formally define the concept of *PO-optimality*. For fully-loaded systems (with 100% utilization), we first propose a PO-optimal scheduler, namely *ASAP-Ensured Earliest Deadline (SEED)*, by focusing on ASAP tasks where the optimality of ALAP tasks' preference is achieved *implicitly* due to the *harmonicity* of the PO-optimal schedules for such systems. Then, for under-utilized systems (with less than 100% utilization), we show the *discrepancies* between different PO-optimal schedules. By extending SEED, we propose a generalized *Preference-Oriented Earliest Deadline (POED)* scheduler that can obtain a PO-optimal schedule for any schedulable task set. The application of the POED scheduler in a dual-processor fault-tolerant system is further illustrated. We evaluate the proposed PO-optimal schedulers through extensive simulations. The results show that, comparing to that of the well-known EDF scheduler, the scheduling overheads of SEED and POED are higher (but still manageable) due to the additional consideration of tasks' preferences. However, SEED and POED can achieve the preference-oriented execution objectives in a more successful way than EDF.

1. Introduction

The real-time scheduling theory has been studied for decades and many well-known scheduling algorithms have been proposed for various task and system models. For instance, for a set of periodic tasks running on a uniprocessor system, the *rate monotonic (RM)* and *earliest-deadline-first (EDF)* scheduling policies are shown to be optimal for static and dynamic priority based preemptive scheduling algorithms, respectively [16]. With the main objective of meeting all the timing constraints, most existing scheduling algorithms (e.g., EDF and RM) prioritize and schedule tasks based only on their timing parameters (e.g., deadlines and periods). Moreover, these algorithms usually adopt the *work conservation* strategy (that is, the processor will not idle if there are tasks ready for execution) and execute tasks *as soon as possible (ASAP)*.

However, there are occasions where it can be beneficial to execute tasks *as late as possible (ALAP)*. For example, to provide better response time for soft aperiodic tasks, the *earliest deadline latest (EDL)* algorithm has been developed to execute periodic tasks at their *latest* times provided that all the deadlines will still be met [8]. By delaying the execution of all periodic tasks as much as possible, EDL has been shown to be optimal where no task will miss its deadline if the system utilization is no more than one [8]. By its very nature, EDL is a non-work-conserving scheduling algorithm: the processor may remain idle even though there are ready tasks. With the same objective, *dual-priority (DP)* was developed based on the phase delay technique [1] for fixed-priority rate-monotonic scheduling [9]. Here, periodic tasks with hard deadlines start at lower priority levels and, to ensure that there is no deadline miss, their priorities are promoted to higher levels after a fixed time offset. Soft aperiodic tasks are executed at the medium-priority level to improve their responsiveness.

Such selectively delayed execution of tasks can be useful for fault-tolerant systems as well. For example, to minimize the overlap between the *primary* and *backup* tasks on different processors (and thus save energy), the execution of backup tasks should be delayed as much as possible [4,12,22]. In fact, EDL has been exploited to schedule periodic backup tasks on the secondary processor to reduce the overlapped execution with their primary tasks for better energy savings [14].

However, when backup tasks (whose primary tasks are on different processors) are scheduled with another set of primary periodic tasks in a mixed manner on one processor [4,12,22], the

* Corresponding author. Tel.: +1 210 458 7453; fax: +1 210 458 4437.
E-mail address: dakai.zhu@utsa.edu (D. Zhu).

http://dx.doi.org/10.1016/j.sysarc.2014.12.001
1383-7621/

execution of backup tasks needs to be postponed as much as possible while the primary tasks should be executed as soon as possible for better performance. Note that, the well-known scheduling algorithms generally treat all periodic tasks *uniformly*. They normally schedule tasks solely based on their timing parameters either at their earliest (e.g., with EDF and RM) or latest times (e.g., with EDL and DP). Hence, neither of them can effectively handle tasks with *different preferences*.

Intuitively, one may consider adopting the hierarchical scheduling approach [18,19] to solve such problems, where tasks with the same preference form a task group and the high-level scheduler would determine only how to schedule different task groups. However, the existing hierarchical scheduling frameworks consider mostly work-conserving algorithms (such as EDF and RM) at both parent and child scheduling components. It is not obvious how such framework can be generalized to non-work-conserving algorithms (such as EDL and DP) in order to comply with tasks' different execution preferences while guaranteeing their timing constraints.

Therefore, we believe that there is a strong incentive to develop effective uniprocessor scheduling algorithms for periodic tasks with different *execution preferences* (e.g., ASAP and ALAP). In addition to fault-tolerant systems, such algorithms can also be applied in mixed-criticality task systems [2], where high-criticality tasks can be given the preference of running early. This makes it possible to discover large amount of slack at earlier time, which could be further exploited to provide better service to low-criticality tasks [20].

However, to the best of our knowledge, such scheduling algorithms have not been well studied in the literature yet. In this work, we consider periodic tasks running on a uniprocessor system where some tasks are preferably executed ASAP while others ALAP. We study effective scheduling algorithms and illustrate their applications. Specifically, the main contributions of this paper are summarized as follows:

- The concept of *preference-oriented (PO)* execution is introduced for tasks with ASAP and ALAP preferences. Two types of *PO-optimal* schedules are defined, where their *harmonicity* and *discrepancies* for fully-loaded and under-utilized systems, respectively, are analyzed.
- An optimal *ASAP-Ensured Earliest Deadline (SEED)* scheduling algorithm, which takes the preference of ASAP tasks into consideration when making scheduling decisions, is proposed for fully-loaded systems.
- A generalized *Preference-Oriented Earliest Deadline (POED)* scheduler is also studied by extending SEED and explicitly managing system idle time, which can obtain a PO-optimal schedule for any schedulable task set.
- The application of the POED scheduler in dual-processor fault-tolerant systems to reduce execution overhead and thus improve system efficiency is further illustrated.
- Finally, we evaluate the proposed schedulers through extensive simulations. The results show that, with manageable scheduling overheads (less than 35 microseconds per invocation for up to 100 tasks), the SEED and POED schedulers can obtain superior performance in terms of achieving tasks' preference objectives when comparing to that of the EDF scheduler. Moreover, the execution overhead in dual-processor fault-tolerant systems can be significantly reduced under POED when compared to the state-of-the-art standby-sparing scheme.

The remainder of this paper is organized as follows. Section 2 reviews closely related work. Section 3 presents system models and some notations. In Section 4, we formally define and investigate the optimality of different preference-oriented schedules. The SEED scheduling algorithm is proposed and analyzed in Section 5. The generalized POED scheduler is addressed in Section 6 and Section 7 illustrates the application of POED in fault-tolerant systems. Section 8 presents the evaluation results and Section 9 concludes the paper.

2. Closely related work

In this section, we review closely related work on scheduling algorithms for periodic real-time tasks running on uniprocessor systems and techniques to reduce execution overhead in fault-tolerant systems. The *earliest-deadline-first (EDF)* and *rate monotonic (RM)* scheduling algorithms, which are well-known optimal schedulers for periodic tasks running on a uniprocessor system, have been studied in [16]. Here, EDF is a dynamic-priority scheduler that prioritizes and schedules tasks based on the deadlines of their current task instances. In comparison, RM is a fixed-priority scheduler that prioritizes tasks according to their periods where tasks with smaller periods have higher priorities. With the objective of meeting all tasks' deadlines, both EDF and RM adopt the *work conservation* strategy, which do not let the processor idle if there are ready tasks, and execute tasks as soon as possible.

For systems that have mixed workload with hard real-time periodic tasks and soft real-time aperiodic tasks, to provide better response time for soft aperiodic tasks, the *earliest deadline latest (EDL)* algorithm has been developed to execute periodic tasks at their latest times [8]. To ensure that there is no deadline miss, EDL considers all instances of periodic tasks within the least common multiple (LCM) of their periods and generate an offline static schedule. For fixed-priority rate-monotonic scheduling, the *phase delay* technique was investigated where the arrival of tasks can be delayed for a certain offset without missing any deadline [1]. Based on this technique, the *dual-priority (DP)* scheme has been developed for rate-monotonic scheduling to improve the responsiveness of soft real-time aperiodic tasks [9].

The idea of delaying the execution of selected tasks has also been exploited in fault-tolerant systems [4,22]. As a common and effective fault-tolerance technique, the *primary/backup (PB)* approach normally schedules multiple copies (i.e., one as *primary* and others as *backup*) of a real-time task on different processors to tolerate a certain number of faults [17]. However, this technique can potentially consume significant system resources (e.g. CPU time and power). Thus, the backup copies are normally canceled as soon as their corresponding primary tasks complete successfully [5]. Hence, to reduce the execution overhead, backup tasks should be scheduled at their latest times to minimize the overlap with their corresponding primary tasks that run on different processors [4,22].

By dedicating one processor as the *spare* for backup tasks, Ejlali et al. studied a novel *Standby-Sparing (SS)* technique for dependent and aperiodic real-time tasks running on dual-processor systems with the goal of saving system energy consumption while tolerating a single permanent fault [10]. Based on the same idea of *separating* tasks on different processors, Haque et al. extended the standby-sparing technique to a more practical periodic task model based on the earliest deadline scheduling [14]. Here, to reduce the overlap between primary and backup copies of the same task, primary and backup tasks are scheduled according to EDF and EDL, respectively, on their dedicated processors [14]. Following this line of research, the fixed-priority (rate-monotonic priority) based standby-sparing scheme was studied in [15]. The generalized standby-sparing schemes for systems with more than two processors were investigated in [13].

Instead of dedicating a processor as the spare, it can be more efficient to allocate primary and backup copies of tasks in a mixed manner on both processors [12]. In this case, on each processor, the