Future Generation Computer Systems 56 (2016) 179–191

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Energy-efficient scheduling of real-time tasks with shared resources☆

Jun Wu

Department of Computer Science and Information Engineering, National Pingtung University, Pingtung 900, Taiwan

HIGHLIGHTS

- This paper explores the DVS scheduling of real-time tasks with shared resources.
- An SRP-based two-speed approach is proposed to work with EDF and RM algorithms.
- The execution speeds of tasks are adjusted dynamically by a blocking-aware method.
- The energy consumption can be reduced without violating tasks' timing constraints.

ARTICLE INFO

Article history:
Received 13 January 2015
Received in revised form
24 April 2015
Accepted 24 May 2015
Available online 4 June 2015

Keywords:
Real-time systems
Dynamic voltage scaling
Task scheduling
Task synchronization
Resource sharing

ABSTRACT

This paper explores the energy-efficient scheduling of real-time tasks on a non-ideal DVS processor in the presence of resource sharing. We assume that tasks are periodic, preemptive and may access to shared resources. When dynamic-priority and fixed-priority scheduling are considered, we use the earliest deadline first (EDF) algorithm and the rate monotonic (RM) algorithm to schedule the given set of tasks. Based on the stack resource policy (SRP), we propose an approach, called blocking-aware two-speed (BATS) algorithm, to synchronize the tasks with shared resources and to calculate appropriate execution speeds so that the shared resources can be accessed in a mutual exclusive manner and the energy consumption can be reduced. Particularly, BATS uses a static low speed to execute tasks initially, and then it switches to a high speed dynamically whenever a task blocks a higher priority task. More specifically, the processor runs at the high speed from the beginning of the blocking until the deadline of the blocked task or the processor becomes idle. In order to guarantee that the deadlines of tasks are met, the static low speed and the dynamic high speeds are derived based on the theoretical analysis of the schedulability of tasks. Compared with existing work, BATS achieves more energy saving because its dynamic high speeds are lower than that of existing work and the processor has less chance to execute tasks at the high speeds. The schedulability analysis and the properties of our proposed BATS are provided in this paper. We also evaluated the capabilities of BATS by a series of experiments, for which we have some encouraging results.

1. Introduction

Nowadays, most modern processors are deployed with dynamic voltage scaling (DVS) technique in order to obtain a better energy efficiency. When a DVS processor is not fully utilized, we can slowdown the execution speeds of tasks such that the overall energy consumption can be reduced. However, this strategy will cause an impact on performance due to the late completion of tasks. Since the late completion of a real-time task is allowed as long as the task meets its deadline, it provides a strong driving force in energy-efficient real-time task scheduling. In the past decades, many excellent approaches have been proposed for energy-efficient real-time task scheduling on DVS processors (comprehensive surveys can be found in [1,2]). Most previous approaches assume that tasks are independent, however, relatively little work has been done when tasks are dependent which is common in many real-life applications.

When dependent real-time tasks are considered, shared resources have to be accessed in a mutually exclusive manner. Existing approaches mainly focus on tasks with non-preemptible critical sections [3–6]. In particular, Zhang and Chanson [3] have proposed a *two-speed strategy* (TSS) based approach, called *dual speed* (DS) algorithm, to execute tasks at the low speed initially and then it switches to the high speed as soon as the tasks are blocked. Later, some TSS-based approaches (e.g., [4–6]) have extended from

☆ An earlier version of this paper was presented at the IEEE 11th International Conference on Embedded Software and Systems (ICESS). The results have been extended in exploring the schedulability analysis as well as the calculation of the execution speeds for dynamic-priority and fixed-priority real-time tasks. More examples, figures, experimental results, and proofs are included in this extension.

E-mail address: junwu@mail.nptu.edu.tw.

http://dx.doi.org/10.1016/j.future.2015.05.012
0167-739X/

this direction. However, making critical sections non-preemptible is not desirable because the blocking time and the number of priority inversions might be increased, and it might result in a higher execution speed than it required for task synchronization.

In this paper, we assume that tasks are preemptible both of critical sections and non-critical sections. An approach, called *blocking-aware two-speed* (BATS) algorithm, is proposed to schedule a set of real-time tasks with shared resources on a non-ideal DVS processor. We use the well-known *earliest deadline first* (EDF) algorithm [7] and the *rate monotonic* (RM) algorithm [7] as the scheduling policies for dynamic-priority and fixed-priority tasks, respectively. Since dependent real-time tasks are considered, BATS uses the well-known *stack resource policy* (SRP) [8] to synchronize the tasks' accesses to shared resources. Under BATS, a task will be executed at a static low speed (calculated off-line) if blocking does not happen, and the processor speed will switch to a high speed (calculated on-line) whenever a blocking occurs. The processor operates at a high speed from the beginning of the blocking until the deadline of the blocked task or the processor becomes idle. Note that BATS calculates the execution speeds based on the theoretical analysis of the schedulability of tasks such that the energy consumption can be reduced without violating the timing constraints of tasks.

The major contributions of this research are two-fold: (1) We explore the DVS scheduling problem under a more realistic task model. Particularly, we assume that tasks are periodic, preemptive (both of critical sections and non-critical sections), and dependent due to concurrent access to shared resources. Furthermore, we consider both of the dynamic-priority and fixed-priority real-time tasks; (2) We explore the schedulability of tasks when they are scheduled by BATS on a non-ideal DVS processor. Related issues of how to derive proper processor speeds and when to change the processor speed between the low and the high speeds are also discussed. Furthermore, the dynamically derived high speeds of BATS are lower than that of the existing work, and the processor has less chance to execute tasks at a high speed. As a result, BATS achieves more energy saving.

The rest of this paper is organized as follows. Section 2 summaries related work in the area of DVS scheduling of real-time tasks. Section 3 defines the system model and the problem. Section 4 proposes BATS with properties and schedulability analysis. Section 5 reports the performance evaluation. Section 6 is the conclusion.

2. Related work

In the literature, there is a simple strategy for scheduling of real-time tasks with shared resources, which uses two speeds, i.e., the low speed and the high speed, for task executions. In general, it executes tasks at the low speed initially and it switches to the high speed as soon as a blocking occurs. As a result, tasks can be scheduled without violating their timing constraints and the energy consumption could be reduced. Such a strategy is called *two-speed strategy* (TSS), and it is common for scheduling of real-time tasks with shared resources, e.g., [3–6,9–13].

By assuming that critical sections are non-preemptible, some TSS-based approaches have been proposed for real-time tasks. Zhang and Chanson [3] first proposed a TSS-based scheduling algorithm for dynamic-priority tasks, called *dual speed* (DS) algorithm. When tasks are scheduled by the well-known *earliest deadline first* (EDF) [7] algorithm, DS calculates the low speed and the high speed based on the sufficient schedulability condition of EDF. Hence, the timing constraints of tasks can be met while energy consumption is reduced. In particular, DS assigns the low speed for executing tasks when they arrive, and the processor speed switches to the high speed as soon as the tasks are blocked. A time interval that the processor is executing at the high speed is called *high speed interval*. Under DS, a high speed interval starts from the beginning of a blocking and ends at the deadline of the blocking task. To achieve further energy saving, Lee et al. [4] explored the same problem with a tighter schedulability analysis, and proposed a multi-speed extension of DS, called *multi-speed* (MS) algorithm. In particular, MS assigns different high speeds for each task by considering different blocking scenarios of task instances. MS reduces more energy consumption than that of DS, and the schedulability of the tasks is still guaranteed.

Later, Elewi et al. proposed two extensions of DS and MS, called *enhanced dual speed* (EDS) algorithm [5] and *improved multi-speed* (IMS) algorithm [6]. EDS and IMS obtain more energy saving by shortening the high speed interval, i.e., they end a high speed interval at the deadline of the blocked task (instead of the deadline of the blocking task) or the earliest time that the processor becomes idle. When critical sections are preemptible, EDS and IMS can work with SRP and *dynamic priority ceiling protocol* (DPCP) [14], respectively, by switching the processor speed to the high speed whenever a critical section is preempted or a blocking occurs. Unfortunately, EDS and IMS are heuristic algorithms and they does not guarantee the schedulability of the tasks.

When critical sections are considered to be preemptible, Jejurikar and Gupta [9] proposed a TSS-based scheduling algorithm for fixed-priority tasks, called *critical section maximum speed* (CSMS), based on the *rate monotonic* (RM) algorithm [7] and the *priority ceiling protocol* (PCP) [15]. CSMS always executes tasks at a low speed and then it switches to the maximum processor speed when tasks being executed in a critical section. Jejurikar and Gupta [16] also proposed an approach for tasks with preemptible critical sections, called *uniform slowdown with frequency inheritance* (USFI), to compute a static speed, called *slowdown factor*, for each task based on the schedulability analysis of tasks with their worst-case blocking time. Under USFI, a blocking task inherits the maximum slowdown factor of the blocked tasks. However, the energy consumption under CSMS and USFI are much higher than that under DS and MS. To achieve more energy saving, researchers also proposed TSS-based algorithms with the slack time stealing method, such as [12]. Recently, Wu and Ke [11] also proposed a more energy-efficient TSS-based approach by considering that critical sections are abortable.

3. System model and problem definitions

3.1. DVS processor models

Nowadays, many modern processors support variable voltage and frequency levels, such as the Intel Core M 5Y70, Intel Core i7 4790K, AMD A8 6410 and Qualcomm Snapdragon 800, such a processor is capable of dynamic voltage scaling and its speed is proportional to the supply voltage. There are two types of DVS processors have been considered in the literature: *ideal* and *non-ideal*. An ideal DVS processor can operate at any speed in the range from the minimum to the maximum available speed, while a non-ideal DVS processor has only discrete speeds. Nowadays, most DVS processors are non-ideal while the ideal DVS processors are only for theoretical analysis purpose. In this paper, we assume that a set of real-time tasks are scheduled on a *non-ideal* DVS processor which supports a set of K discrete speeds $\mathcal{S} = \{s_1, s_2, \ldots, s_K\}$, where $s_1 < s_2 < \cdots < s_K$. Let s_{min} and s_{max} denote the minimum and the maximum speeds (i.e., $s_{min} = s_1$ and $s_{max} = s_K$), respectively. Without loss of generality, we assume the s_{max} is 1 and all other speeds are normalized with respect to the maximum speed s_{max}.

In the literature (such as [17,18,16,19,3,5,6,20]), the power consumption of a DVS processor is modeled as a function of its operation speed. Let $PC(s)$ be the power consumption function, where