



Slack computation for DVS algorithms in fixed-priority real-time systems using fluid slack analysis

Da-Ren Chen ^{*}

Department of Information Management, National Taichung Institute of Technology, 111 Gong Jhuan Rd., Chung Ho, Taipei, Taiwan, ROC

ARTICLE INFO

Article history:

Received 14 September 2009
Received in revised form 24 September 2010
Accepted 27 September 2010
Available online 15 October 2010

Keywords:

Real-time systems
Fixed-priority scheduling
Dynamic voltage scaling
Slack time analysis

ABSTRACT

This work presents a scheduling algorithm to reduce the energy of hard real-time tasks with fixed priorities assigned in a rate-monotonic policy. Sets of independent tasks running periodically on a processor with dynamic voltage scaling (DVS) are considered as well. The proposed online approach can cooperate with many slack-time analysis methods based on low-power work demand analysis (lpWDA) without increasing the computational complexity of DVS algorithms. The proposed approach introduces a novel technique called low-power fluid slack analysis (lpFSA) that extends the analysis interval produced by its cooperative methods and computes the available slack in the extended interval. The lpFSA regards the additional slack as *fluid* and computes its length, such that it can be moved to the current job. Therefore, the proposed approach provides the cooperative methods with additional slack. Experimental results show that the proposed approach combined with lpWDA-based algorithms achieves more energy reductions than do the initial algorithms alone.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Dynamic voltage scaling (DVS) is a standard technique for managing the power consumption of systems [21]. A DVS processor can vary its operating frequency and voltage during runtime to use the quadratic relationship between energy consumption and supply voltage of CMOS technology. In recent years, computations and communication have moved steadily toward mobile and portable devices with limited power supply. Therefore, many primary IC producers have developed modern processors with DVS capability, including Intel's XScale[®] [10], the mobile Athlon[®] by AMD [1] and SamSung's Cortex[®] [19].

Many studies have focused on scheduling real-time applications on DVS processors [2,5,6,8,9,11,12,17,18,20]. The developed approaches differ in many aspects, such as on-line and off-line scheduling algorithms, handling discrete/continuous voltage levels, assuming average-case execution time (ACET), best-case execution time (BCET), or worst-case execution time (WCET) of each task, allowing intra-task/inter-task voltage transitions, and assuming fixed/dynamic priority assignment. However, these approaches have a common objective and encounter the same difficulties. Because reducing the supply voltage decreases max-

imum achievable clock speed [15], most DVS algorithms for real-time systems reduce supply voltage dynamically to the lowest possible level while satisfying the soft/hard timing constraints of a system. To satisfy the timing constraints of real-time tasks, DVS technique can utilize slack times when adjusting voltage levels. Consequently, the energy efficiency of a DVS algorithm markedly depends on the accuracy of computing available slack.

Many previous studies have investigated slack time analysis [5,8,11,12,14,17] while assuming a feasible schedule. Pillai and Shin [17] proposed a cycle-conserving rate-monotonic (ccRM) scheduling scheme that contains off-line and on-line algorithms. The off-line algorithm computes the worst-case response time of each task and derives the maximum speed needed to meet all task deadlines. It recomputes the utilization by comparing the actual time for completed tasks with WCET schedule. In other words, when a task completes early, they have to compare the used actual processor cycles to a pre-computed worst-case execution time schedule. This WCET schedule is also called *canonical schedule* [2] whose length could be the least common multiplier of task periods. ccRM is a conservative method, as it only considers possible slack time before the next task arrival (NTA) of current job. Gruian proposed a DVS method for off-line task stretching and on-line slack distribution [8]. The off-line part of this method consists of two separate techniques. One focuses on the intra-task stochastic voltage scheduling that employs a task-execution length probability function. The second technique computes stretching factors by using a response time analysis. It

^{*} Tel.: +886 2 8941 5143x117; fax: +886 2 8941 5142.

E-mail addresses: danny@cc.hwh.edu.tw, danny063@gmail.com

is similar to Pillar and Shin's off-line technique, but instead of adopting a stretching factor for all tasks that before NTA, Gruian assigns different stretching factor to the individual task within the longest task period. Kim et al. [12] proposed a greedy on-line algorithm called the low-power work-demand analysis (lpWDA) that derives slack from low-priority tasks, as opposed to the method in [8,17] that gains slack time from high-priority tasks. This algorithm also balances the gap in voltage levels between high-priority and low-priority tasks. Its analysis interval limited by the longest of task periods is longer than NTA. Thus, lpWDA gains more energy saving than the previous rate-monotonic (RM) DVS schemes applying NTA. Many slack time analysis methods considered additional assumptions [5,9,14,16]. Kim et al. proposed a preemption-aware DVS algorithm based on lpWDA, which is composed of *accelerated-completion* (lpWDA-AC) and *delayed-preemption* (lpWDA-DP) techniques to decrease the preemption times of DVS schedules [14]. lpWDA-AC attempts to avoid preemption by adjusting voltage/clock speed, such that it is higher than the lowest possible values computed using lpWDA. lpWDA-DP postpones preemption points by delaying an activated high-priority task as late as possible while guaranteeing a feasible task schedule. Both techniques reduce energy consumption more than the initial ccRM and lpWDA techniques on the assumption of context-switching overhead. Mochocki et al. [16] also proposed a transition-aware DVS algorithm for decreasing the number of voltage/speed adjustments, called the low-power limited-demand analysis with transition overhead (lpLDAT) scheme, which accounts for both time and energy transition overhead. Its algorithm computes an efficient speed level based on average-case workload; notably, this speed can be used as a *limiter*. If the *limiter* is higher than the speed predicted by lpWDA, lpLDAT knows that lpWDA is being too aggressive and applies the limiter to the present schedule. On the assumption of transition overhead, this technique with slack time analysis also saves considerable energy when compared with that by the previous methods. He and Jia [9] developed a fixed-priority scheduling with threshold (FPPT) scheme that eliminates unnecessary context switches, thereby saving energy. FPPT assigns each task a pair of predefined priority and corresponding preemption threshold. He and Jia applied a novel algorithm to compute a static slowdown factors by formulating the problem as a linear optimization problem. In addition, they considered energy consumption of a task set under different preemption threshold assignments. Chen and Hsu [5] proposed a tree structure corresponding to a set of pinwheel tasks [15]. They also proposed a DVS (the low-power job contiguous real-time scheduling, lpJCRT) algorithm, which manipulates a tree structure to distribute available slack evenly among other tasks.

Experimental results obtained by Kim et al. [12] indicated that recent DVS algorithms for fixed-priority real-time tasks are less efficient than that of dynamic-priority tasks, leading to more improvements for a better DVS method. The main reason for energy inefficiency of RM DVS scheduling is that, in RM schedules, priority-based slack-stealing methods do not work as efficiently as they do in earliest-deadline first (EDF) scheduling [12]. However, in EDF schedules, high-priority tasks play an efficient slack distributor of tasks because their slack can be utilized fully by tasks starting before NTA. Therefore, the energy saving achieved by EDF scheduling algorithms, such as that by the ccEDF [17], DRA and AGR [2] is close to the theoretical lower bound [13].

This work improves the on-line slack computation capability of RM DVS algorithms. Based on existing RM DVS algorithms, we propose an on-line slack-time computation scheme using *fluid slack analysis*, which computes the length of potential slack in an interval longer than the longest of task periods. The pro-

posed method does not need to compute or perform a simulation for stochastic data, which varies according to different applications. With a slight modification, lpFSA can be applied to many RM DVS scheduling scheme with various assumptions, including transition and preemption criteria. Moreover, the proposed algorithm has a time complexity of $O(n)$ where n is the number of tasks. Therefore, it does not increase computational complexity of the existing on-line DVS algorithms. Experimental results indicate that existing RM DVS algorithms combined with the proposed method can reduce energy consumption by 5–25% compared with that by initial algorithms such as lpWDA, lpLDAT, etc.

The remainder of this paper is organized as follows. Section 2 explains the motivation of this work. The basic idea of fluid slack analysis is proposed in Section 3. Section 4 describes the proposed technique and algorithm. Section 5 provides theorems to prove the schedulability of lpFSA as well as lpWDA. We present the performance evaluation in Section 6. Section 7 gives conclusions and the directions for future work.

2. Preliminaries

This section first presents the assumptions of task systems and introduces the necessary notations. This work focuses on how to estimate additional slack for existing RM DVS scheduling schemes. Many slack-time analysis techniques with different purposes (e.g., transition-aware and preemption-aware schemes) can utilize lpFSA easily; throughout this paper, these techniques are called *host* algorithms of lpFSA. This section also outlines the ideas underlying the lpWDA and lpLDA algorithms. Other techniques, such as the lpWDA-AC, lpWDA-DP [14] and lpLDAT [16] techniques, are abridged.

2.1. System model

This work considers preemptive hard real-time systems in which periodic real-time tasks are scheduled under an RM scheduling policy. The DVS processor used in the model operates at a finite set of supply voltage levels $V = \{v_1, \dots, v_{max}\}$, each with an associated speed. Processor speed is normalized by $S_{max} = 1$ corresponding to $v_{max} = 1$, yielding a set $S = \{s_1, \dots, 1\}$ of speed levels. A set of n periodic tasks is denoted by $T = \{\tau_1, \tau_2, \dots, \tau_n\}$, where the tasks are assumed mutually independent. Each task τ_i is described by its worst-case execution cycles wc_i , and average case execution cycles ac_i ($wc_i \geq ac_i$). Throughout this paper, the execution cycles of each task are called *work* for short. Additionally, each task τ_i has a shorter period length p_i (i.e., a higher priority) than that of τ_j when $i < j$, and p_n is the longest of task periods. The relative deadline d_i of τ_i is assumed equal to its period length p_i . Each task is invoked periodically by a *job*, and the k th job of task τ_i is $\tau_{i,k}$. The first job of each task is assumed activated at time $t = 0$. Each job is described by a release time, $r_{i,j}$, deadline, $d_{i,k}$, and number of cycles that have been executed $ex_{i,k}^j$. The utilization U of a task set T is denoted by $\sum_{\tau_i \in T} \frac{wc_i}{p_i}$. During run-time, we refer to the earliest job of each task not completed as the *current* job for that task, and that job is indexed with *cur*. The deadline of the current job for task τ_i is d_i^{cur} , and ex_i^{cur} denotes the number of cycles that the current job of τ_i has executed.

Without loss of generality, when τ_i is the first scheduled task after time $r_{n,k-1}$, where $i \neq n$, the **border** is the next release time of τ_n (i.e., the $r_{n,k}$). In the fluid slack analysis method, available slack in the interval [border, $r_{n,k+1}$) is estimated.

متن کامل مقاله

دریافت فوری ←

ISIArticles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات