Journal of Systems Architecture 56 (2010) 352–367

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Transition-aware DVS algorithm for real-time systems using tree structure analysis

Da-Ren Chen [a,*], Chiun-Chieh Hsu [b], You-Shyang Chen [a], Chi-Jung Kuo [b,c], Lin-Chih Chen [d]

[a] Department of Information Management, Hwa Hsia Institute of Technology 111 Gong Jhuan Rd., Chung Ho, Taipei, Taiwan, ROC
[b] Department of Information Management, National Taiwan University of Science and Technology, #43, Sec.4, Keelung Rd., Taipei 106, Taiwan, ROC
[c] Department of Information Management at Technology and Science Institute of Northern Taiwan, Taipei, Taiwan, ROC
[d] Department of Information Management, National Dong Hwa University, Hualien, Taiwan, ROC

ARTICLE INFO

Article history:
Received 30 September 2009
Received in revised form 1 May 2010
Accepted 4 May 2010
Available online 13 May 2010

Keywords:
Dynamic voltage scaling
Real-time scheduling
Distance-constraint tasks scheduling

ABSTRACT

Dynamic voltage scaling (DVS) is a key technique for embedded real-time systems to reduce energy consumption by lowering the supply voltage and operating frequency. Many existing DVS algorithms have to generate the canonical schedules or estimate the lengths of slack time in advance for generating the voltage scaling decisions. Therefore, these methods have to compute the schedules with exponential time complexities in general. In this paper, we consider a set of *jitter-controlled*, independent, periodic, hard real-time tasks scheduled according to preemptive pinwheel model. Our approach constructs a tree structure corresponding to a schedule and maintains the data structure at each early-completion point. Our approach consists of *off-line* and *on-line* algorithms which consider the effects of transition time and energy. The off-line and on-line algorithm takes $O(k + n \log n)$ and $O(k + (p_{max}/p_{min}))$ time complexity, respectively, where n, k, p_{max} and p_{min} denotes the number of tasks, jobs, longest and shortest task period, respectively. Experimental results show that the proposed approach is effective in reducing computational complexity, transition time and energy overhead.

1. Introduction

Power-aware computing has widely spread not only for mobile devices powered by batteries, but also for large systems in which the cost of energy consumption and cooling is substantial. With dynamic voltage scaling (DVS) technique [2,17,20,21,23,25,26,28,32,33,39,48], processors perform at a range of voltage and frequencies. Since the energy consumption E of CMOS circuits has a quadratic dependency on the supply voltage, lowering the supply voltage is one of the most effective ways to reduce the energy consumption while satisfying the time constraint of each real-time task.

Most DVS scheduling algorithm are known to be quite effective in reducing the energy consumption of a target system. However, a significant limitation of DVS processor is that they are unable to change the operation voltage and frequency immediately. The limitation, known as transition time overhead, includes translation time required for power delivery system and the time for relocking a clock generator requires approximately 130 μs in Intel® M processors [13,49]. The architecture, in addition, needs 10–15 μs to ensure system memory access unavailability to match isochronous device needs. Therefore, ignoring time overhead in real-time systems may incur deadline misses, which results in low system performance or even system failure [32]. Another problem is the transition energy overhead, which increases the systems' energy consumption if DVS algorithm adjusts voltage/speed level frequently. Therefore, much literature ignoring transition energy overhead may not reduce actual total energy consumption in the systems.

In the network systems, *jitter* or *packet delay variation* (PDV) is defined as the variation in the time between successive packet arrivals caused by network congestion, time drift or route changes [28,47]. PDV is an important quality-of-service factor to evaluate the network performance. One of the most widespread techniques to improve PDV is pinwheel scheduling [16,24,27,29,31,35]. A pinwheel task i is characterized by two positive integer parameters an execution requirement and a window length with explanation that the task i need to be allocated the shared resource for at least a out of every b consecutive time units. Pinwheel task systems are the first motivated by the performance requirements of satellite-based communications. In the recent applications, broadband 3G (B3G) wireless communication systems provide a packet-switched core network to support broadband wireless multimedia services. The resource management policies in the cell of B3G system are to guarantee the quality-of-service (QoS) of real-time (RT) traffics. To guarantee the QoS of RT traffics in a cell, many researchers [4,24,30,31] proposed the pinwheel scheduling algorithms to reduce the jitter of variable bit rate (VBR) traffic in a cell. In addition, pinwheel scheduling is also applied in the channel assignment policies with buffer and preemptive priority for RT

* Corresponding author. Tel.: +886 2 8941 5143; fax: +886 2 8941 5142.
E-mail addresses: danny@cc.hwh.edu.tw, danny063@gmail.com (D.-R. Chen).

1383-7621/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2010.05.003

traffics. In other applications, such as the medium access control (MAC) layer of CDMA and TDMA-based wireless networks [8,10,28], many pinwheel scheduling schemes are proposed for solving the frame-based packet scheduling problems. These pinwheel methods provide low delay and low jitter for RT traffic and short-queue length for non-RT traffic.

In the periodic real-time systems, the *jitter* is defined as the difference in the end-to-end delay between successive jobs in a sequence with respect to a processor. It is usually caused by the changes of task periods or the interference of other tasks [33]. The former applications using jitter-controlled or pinwheel algorithm are all designed for a single node. Pinwheel scheduling is also used in the end-to-end scheduling model [15] to solve the timing and reliability constraints. For example, in the wireless sensor network applications, a multi-sensor situation assessment task is invoked either periodically or triggered by certain events. This assessment process may have one or more input tasks to collect data from different sensors. The end-to-end deadline between a trigger event and the corresponding response processing to response may have somewhere limitations. Similar requirements exist in the phased-array radar systems; the dwell tasks collect device data and must recognize properties of the aircrafts within a certain end-to-end deadlines [15]. Other examples include the use of broadcast disks [4] in the on-board automotive navigational systems [4,18,19] that allow for automated route guidance and automatic routing around traffic incidents.

Many applications have applied the concept of jitter control in their computer and communication hardware/software systems. For example, the delay and jitter control in ATM (Asynchronous Transfer Mode), a multimedia stream requires QoS including end-to-end delay 100 ms in the network systems. Novel real-time applications like Voice over IP (VoIP) and Video on Demand (VoD) have jitter control requirements. In many RT applications, tasks must be executed in a distance- constrained manner, rather than just periodically. In the distance-constrained task systems (DCTS), the temporal distance between any two consecutive executions of a job should always be less than a certain value. In DCTS, pinwheel tasks transform the distance-constraints into 2^n multiples of other shorter periods [7,14], which are shorter or equal to their original distance-constraints. The advantage of the period transformation is that the produced schedules have regular start, preemption and finish times, and therefore provide good predictability.

2. Related work

Many previous works studied the problem of real-time scheduling with inter-task DVS [2,20,32,37,39,42,43,45]. Yao et al. [46] proposed a theoretical DVS model and an $O(n^3)$ algorithm for computing minimum energy DVS schedule in a continuous variable voltages processor. Ishihara et al. [17] developed an optimal voltage allocation method using a discrete variable voltage processor. The optimality of this technique is confined to a single task. Kwon et al. [25] presented an optimal discrete approach, which combined with the continuous version in [46] and therefore requires $O(n^3)$ time. Recent work proposed by Li and Yao [26] gives $O(dn \log n)$ time algorithm that constructs a minimum energy schedule without computing the optimal continuous schedule in advance. Nevertheless, they still have to generate a standard schedule prior to voltage scaling algorithm. Therefore, the lengths of such schedules depend on the LCM of the length of task periods and cannot be completed in polynomial time. These off-line techniques, in addition, considering the early-completed tasks cannot utilize the produced slack effectively to save more energy. Moreover, they did not consider inter-task transition overhead, which likely cause deadline misses or even system failure.

In practice, real-time systems usually reveal large variations in the implementation workload experienced by their applications [2,11]. Many researches [2,21,22,32,34], therefore, focused on the dynamic reclaiming techniques instead of simply the static power managements [45,46]. Krishna and Lee [22] presented a power-aware scheduling method for cyclic and periodic models with two voltage levels. Their *speculative* speed-reduction only considered the tasks with identical deadline. Aydin et al. [2,3] proposed a dynamic reclaiming algorithm (DRA for short) which detects an early-completed task and decreases the continuous voltage level of the successor tasks in order to save more energy while meeting their deadlines. DRA has to generate a *canonical* schedule beforehand and ignores the transition overhead. In addition, they proposed an aggressive speed-reduction method (AGR for short) [3] that considers a speculation speed level when early-completion tasks are anticipated and the execution speed can be reduced aggressively. However, this speculation level, such as average workload, is suggested by a statistical value, which assumes the future tasks will most probably present an execution requirement far and away the worst-case execution time (WCET). It probably causes the frequent fluctuations of voltage/speed scaling especially when a job approaches its deadline and harmful to energy efficiency. Kim et al. [21] developed a method called *lpWDA* that uses a greedy on-line algorithm in RM scheduling to estimate the amount of available slack and apply it to the current job. At each time, only one job stretches its execution by using the estimated slack derived from the higher or lower priority tasks. Since the slack cannot be shared with other jobs, it increases the number of voltage/speed transitions. Since *lpWDA* is often too aggressive, it increases the transition overhead and the fluctuant voltage scaling which are unfavorable to energy saving [2,45]. In addition, they assumed that a DVS processor changes its voltage level within a continuous range and did not consider the transition overhead. Mochocki et al. [32] proposed an on-line technique with discrete voltage scaling for accounting for transition overhead in preemptive fixed-priority task systems. They compute an efficient speed based on the average-case workload and highly rely on the estimative speed. However, the early-completion behavior of each job is difficult to predict because the execution requirement of the real-time job changes irregularly [12]. In addition, in the dynamic systems in which tasks join or leave the systems frequently, these approaches could not provide timely information to reach the ideal energy efficiency.

In [44], Kumar et al. proposes a DVS algorithm based on the assumptions that non-preemptive tasks have precedence constraints. In their system model, they considered both computation and communication workloads in the network with real-time constraints. They proposed an on-line method called *distributed slack propagation* (DSP) algorithm, which performs slack allocation across tasks and message without violating the deadline and precedence constraints. When a dynamic slack is generated on a processor, they have to decide whether the current task should utilize the slack or not. In order to reach better energy gains, they introduced a data structure called *slack propagation tree* (SPT), where each message and task is represented as a separate node in the tree. SPT is constructed for deciding how to use the slack generated by current early-completion tasks. The run-time complexity of the DSP algorithm can be estimated as $O(v^2(d_t + d_m))$ where v^2, d_t and d_m denotes the number of task/message, speed level of DVS processor and modulation level of wireless emitter, respectively.

There are also many DVS scheduling techniques for independent jobs that consider transition overhead. Seo et al. [40] proposed an optimal intra-task voltage scaling method that considers transition overhead during compiling time. However, it cannot be directly applied to the inter-task voltage scaling systems. They also combined inter-task and intra-task in [41] with the EDF-based