BTS-SRP: An Energy-Efficient Concurrency Control Protocol for Embedded Real-Time Systems

Jun Wu and Jun-Xing Wu
Department of Computer Science and Information Engineering, National Pingtung Institute of Commerce
900 Pingtung City, Taiwan, R.O.C.
junwu@npic.edu.tw and s100318010@student-mail.npic.edu.tw

Abstract—**We explored the scheduling problem of dependent real-time tasks that may access multiunit resources on a non-ideal dynamic voltage scaling (DVS) processor. Based on the stack resource policy (SRP) protocol and the earliest deadline first (EDF) algorithm, we propose an approach, called blocking-time stealing stack resource policy (BTS-SRP), for the scheduling of dependent real-time tasks. Under the BTS-SRP, tasks are executed at proper processor speeds which are calculated according to the sufficient schedulability condition of the EDF and the SRP. In order to obtain more energy saving, a blocking-time stealing method is also proposed to dynamically adjust the processor speed. Our experimental results show that the BTS-SRP outperforms pervious work.**

Keywords-Real-Time Task Scheduling; Concurrency Control; Dynamic Voltage Scaling

I. INTRODUCTION

In the past decades, many excellent approaches have beenproposed so as to reduce energy consumption of real-timetasks on DVS platforms. A comprehensive survey ofenergyefficient real-time scheduling can be found in [1]. Most worksassume that tasks are independent, however, relatively little work hasbeen done for dependent real-time tasks. In many real applications, however, tasks are dependent because of resource sharing.

Based on the *priority ceiling protocol* (PCP)[2] and/orthe *stack resourcepolicy* (SRP)[3], a simple strategy that uses two speeds, i.e., low speed and high speed, to execute tasks. Initially, a task is executed at the low speed and then it switches to the high speed when it is blocked. Those two speeds are usually calculated based on the sufficient schedulability condition of tasks so that the energy consumption can be reduced without violating their timing requirements. This strategy is called two-speed strategy (TSS), and it is common for energy-efficient scheduling of dependent tasks under different assumptions on the task and system models, such as [4-11].In particular, Zhang and Chanson [4], and,Jejurikar and Gupta [7]were proposed excellent TSS-based approaches for dependent tasks with non-preemptible and preemptible critical sections, respectively.

This paper considers that tasks are periodic and preemptible, and dependent due to the resource sharing. We assume that a set of multiunit resources can be accessed during the execution of tasks. Note that each multiunit resource has a fixed number of units in the system. An SRP-based approach, called *blocking-time stealing stack resource policy* (BTS-SRP), is proposed so as to manage the resource sharing problem and to reduce the energy consumption of tasks without violating their timing constraints. The BTS-SRP uses the EDF for scheduling dynamic-priority tasks. Under the BTS-SRP, tasks are scheduled to be executed at proper processor speeds which are calculated according to the sufficient schedulability condition for the EDF algorithm. We also propose a blocking-time stealing method to reclaim the blocking time for slowing down the processor speed. As the results, more energy saving could be obtained. The capabilities of the BTS-SRP were evaluated by experiments. It shows that our proposed BTS-SRP outperforms previous work.

II. SYSTEM MODEL AND PROBLEM DEFINITIONS

A. DVS Processor Models

We assume that tasks are executed on a *non-ideal* DVS processorwhich supports a set of k discrete speeds $S = \{s_1, s_2, \dots, s_k\}$, where$s_1 < s_2 < \cdots < s_k$. Let s_{min} and s_{max} denote the lowest and the highest speeds (i.e., $s_{min} = s_1$ and $s_{max} = s_k$). All speeds are normalized with respect to the s_{max}=1. The power consumption of a DVS processor is defined as a function of the processor speed, denoted by $PC(s)$.Let $s(t)$ be the processor speed at time t. The energy consumption $EC_{(t_1,t_2]}$ in time interval $(t_1, t_2]$ is defined by$\int_{t_1}^{t_2} PC(s(t))\, dt$.

B. Task and Resource Models

A set of n periodic dependent tasks$\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_n\}$ is considered in this paper. A periodic task $\tau_i$is a template of its instances. The task instances of a task will arrive regularly for every periodT_i. Let $\tau_{i,j}$ denote the *j*th instance of task τ_i. The worst-case computation amount and the deadline of a task τ_i are defined by C_i andD_i. When τ_i is executed at a speed s_x, the worst-case execution time of τ_i isC_i/s_x. We consider well-formed tasks which satisfy$0 \le C_i \le D_i \le T_i, \forall\, \tau_i \in \mathcal{T}$. We also assume that the deadline is equal to the period, i.e., $D_i = T_i$. The priority of a taskτ_i is defined by p_i.

We assume that a set of *m*multiunit resources $\mathcal{R} = r_1, r_2, \dots, r_m\}$ are accessed by tasks[3].Each multiunit resource r_j has a fixed number of units, denoted as N_{r_j}. A task τ_i may make one or more requests for accessing multiunit resources during its execution. Let $\mu_{r_j}(\tau_i)$be the number of units of resource r_j requested by task τ_i, and $\mu_{r_j}(\tau_i) < N_{r_j}$ for $1 < i < n$. Resources are assumed to be guarded by semaphores, and the time interval during the accessing of a resource is called a *critical section*.

Let $\mathcal{Z}_i =< z_{i,1}, z_{i,2}, \dots, z_{i,n_i} >$ be the list of τ_i's critical sections. Each critical section $z_{i,j}$ is a request which needs $\mu(z_{i,j})$ units of the resource $\mathcal{R}(z_{i,j})$. The computation amount of a critical section $z_{i,j}$ is defined by $|z_{i,j}|$. The execution time of a critical section $z_{i,j}$ is $|z_{i,j}| / s_x$ if the processor speed is s_x .Before a task can enter a critical section, it must wait for sufficient units of the resource and the access right has been granted. A task τ_i is said to be *blocked* by a lower-priority task τ_j's critical section if it has to wait for τ_j to exit the critical section in order to resume its execution.

C. Problem Definitions

Let *lcm* denotes the *least common multiple* of all tasks' periods (also called *hyperperiod*). Since the taskset $\mathcal{T}$ repeats an identical execution trace every hyperperiod, we only need to examine the time interval (0, *lcm*] for analyzing the performance and schedulability of the entire schedule [13]. The research problem is as follows:

For a given set of dependent real-time tasks $\mathcal{T}$ and a set ofshared multiunit resources $\mathcal{R}$. The problem is to schedule $\mathcal{T}$ and to synchronize their accesses of shared multiunit resources $\mathcal{R}$ on a non-ideal DVS processor such that

(1) tasks have to meet their timing constraints, and

(2) minimizes $\int_0^{lcm} PC(s(t))\,dt$. □

III. BLOCKING-TIME STEALING STACK RESOURCE POLICY (BTS-SRP)

A. Task Scheduling and Resource Access Control

The rules for task scheduling and concurrency control are as the same as those of the EDF and the SRP, respectively. Let π_i be the preemption level of a task τ_i. Under the BTS-SRP, each task is assigned a fixed preemption level inversely proportional to its deadline (i.e., $\pi_i > \pi_j \Leftrightarrow D_i < D_j$).

Each resource r_j is required to have a current ceiling CL_{r_j}, which is calculated as a function of the units of r_j that are currently available. It can be computed by

$$CL_{r_j} = \max_{\tau_i \in \mathcal{T}}\{\{0\} \cup \{\pi_i : n_{r_j} < \mu_{r_j}(\tau_i)\}\}$$

,where n_{r_j} is the number of units of r_j which are currently available. We also define π_s as the system ceiling which is computed as follows:

$$\pi_s = \max_{r_j \in \mathcal{R}}\{0, CL_{r_j}(n_{r_j})\}$$

The rules for concurrency control of the BTS-SRP are as the same as those of the SRP. Note that we use $t^h_{i,j}$ to denote the time that $\tau_{i,j}$ becomes the highest-priority task.

B. Schedulability and the Base Processor Speed

Under the BTS-SRP, the execution speeds of tasks are considered based on the worst-case conditions for the EDF algorithm. Equation (1) is the sufficient schedulability condition for the EDF and the SRP.

$$\sum_{k=1}^{n} \frac{C_k + B_k}{D_k} \tag{1}$$

,where B_i is the maximum blocking time of τ_i. Note that the worst-case blocking time of τ_i is B_i/s_x when the processor speed is s_x. The following equation shows the value of the *base processor speed* s^b which is the lowest processor speed for executing tasks without violating their timing constraints.

$$s^b = \max_{s_j \in S.}\{s_j : \sum_{k=1}^{n} \frac{C_k + B_k}{D_k} \le s_j\} \tag{2}$$

C. Task Execution Speed and Blocking-Time Stealing

Firstly, all task's critical sections are assigned to be executed at the base processor speed s^b. It ensures that the actual blocking time of any task τ_i does not exceed B_i/s^b. Secondly, every task instance $\tau_{i,j}$ is assigned to be executed at speed $s^*_{i,j} \le s^b$ excepts its critical sections (note that all critical sections are executed at the base processor speed).

The *blocking interval* of a task instance is a time interval when the task instance is blocked. Since the BTS-SRP extends from the SRP, a task instance has at most one blocking interval before it starts [3]. For any task instance $\tau_{i,j}$, the blocking interval is starting from the time that $\tau_{i,j}$ becomes the highest-priority task among all tasks ready to run, i.e., at time $t^h_{i,j}$. And the interval is ending at the time the task instance is scheduled to start its execution. If $\tau_{i,j}$ starts its execution at time $t^s_{i,j}$, then the blocking interval is $[t^h_{i,j}, t^s_{i,j})$. Since all critical sections are executed at s^b, the worst-case blocking time of τ_i is B_i/s^b, and $t^s_{i,j} - t^h_{i,j} \le B_i/s^b$.

For any task τ_i, its computation amount C_i consists of two parts: cC_i and nC_i, where cC_i is the total computation amount of its critical sections, i.e., $cC_i = \sum_{z_{i,j} \in \mathcal{Z}_i} |z_{i,j}|$, and nC_i is the computation amount of its non-critical part, i.e., $nC_i = C_i - cC_i$.

When the actual blocking time is less than the worst-case blocking time, we get additional $B_i/s^b - (t^s_{i,j} - t^h_{i,j})$ time for executing the task instance $\tau_{i,j}$.This additional time is adopt to slow down the execution speed from s^b to $s^*_{i,j}$ so that energy consumption could be reduced. We called this method as *blocking-time stealing*.

According to the BTS-SRP, the execution time of a task instance $\tau_{i,j}$ is $nC_i/s^*_{i,j} + cC_i/s^b$. For ensuring the schedulability of all tasks, $\tau_{i,j}$'s execution time cannot exceed $C_i/s^b + B_i/s^b - (t^s_{i,j} - t^h_{i,j})$. Hence, the processor speed $s^*_{i,j}$ can be calculated as follows:

$$s^*_{i,j} = \max_{s_k \in S} \{s_k : \frac{s^b n C_i}{B_i - s^b(t^s_{i,j} - t^h_{i,j}) + nC_i} \le s_k\} \qquad (3)$$

IV. PERFORMANCE EVALUATION

We have implemented a simulation of a DVS environment to schedule different task workloads. In our simulation, the speeds of the processor are from 0.05 to 1 increased by 0.05. The performance of our proposed BTS-SRP is compared with the following approaches: *uniform slowdown with frequency inheritance* (USFI) [7], *independent task set transformation* (ITST), *BTS-SRP without blocking-time stealing* (BS) and *maximum speed* (MS). Where ITST transforms the given tasks into independent tasks and uses the EDF algorithm to schedule the transformed tasks. The BS schedules tasks to be executed at the base processor speed s^b. The MS is a baseline approach which schedules tasks to be executed at the s_{max}under the EDF and the SRP.

A. Performance Metrics and Data Set

The primary performance metric of interest is the energy consumption of tasks whichis the sum of the energy consumption of every task instance executed during the simulation time. We assume that the power consumption function of processor speeds be $PC(s_i) = (0.08 + 1.52v_i^3)$Watts [14]. We set the supply voltage $v_i = s_i * 10$, $\forall s_i \in S$. The energy consumption in time interval $[t_1, t_2)$ can be obtained by $\int_{t_1}^{t_2} PC(s(t))\, dt$.

For generating feasible task sets, we set the utilization of tasks from 0.2 to 1 with an increment of0.1. The period of a task was selected form 100 to2000by normal distribution. The worst-case computation amount of task was selected form 10 to 300 by normal distribution. The number of shared multiunit resources is 5 to 10 and the number of units foreach resource is 1 to 5.

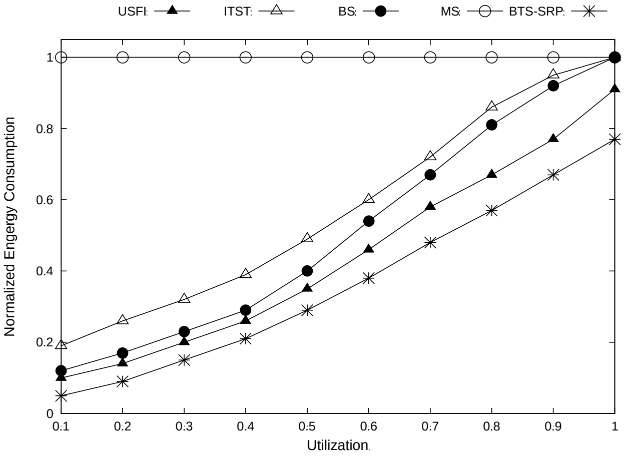

Figure 1. Normalized energy consumption.

For each task τ_i, we also set the *resource usage ratio* be 0.3, which is defined as cC_i/C_i. The simulation time is 100,000 and over 10 task sets per utilization factor were tested in the simulation and the results are averaged.

B. Simulation Results

Figure 1 shows the experimental results of different approaches. The energy consumption of the USFI, ITST, BS, and our proposed BTS-SRP are normalized with respect to the baseline approach, i.e., MS (The tasks are always executed ats_{max}.). Figure 1 shows that our proposed BTS-SRP outperforms all others.The performance ranking is MS, ITST, BS, USFI, and BTS-SRP (from the worst to the best). The performance of the MS is the worst because tasks are executed at s_{max}. The ITST and the BS schedule tasks to be executed at the speed $\min_{s_j \in S}\{ s_j \mid \sum_{i=1}^{n} \frac{C'_i}{D_i} \le s_j\}$ and the base processor speeds^b, respectively. The BS outperforms the ITST because the ITST transforms dependent tasks into independent tasks by adding the worst-case blocking time to tasks' computation. In other words, it assumes the blocking will be occurred for every task instances. When the actual blocking time is less than the worst-case blocking time, the BS outperforms the ITST. The performance of the USFI is better than others excepts our proposed BTS-SRP. It is because the blocking-time stealing method is employed by the BTS-SRP so that more energy saving could be achieved.

V. CONCLUSION

In this paper, we propose an approach, called blocking-time stealing stack resource policy (BTS-SRP), to schedule dependent real-time tasks and to assign proper processor speed for their executions on a non-ideal DVS processor. The blocking-time stealing method can dynamically adjust the processor speed such that more energy saving could be obtained. The capabilities of our proposed approach were evaluated by experiments. It is shown that the BTS-SRP outperforms others.

ACKNOWLEDGMENT

This paper is supported in part by a research grant from the National Science Council (NSC) of Taiwan under the grant number NSC-101-2221-E-251-005.

REFERENCES

[1] J.-J. Chen and C.-F. Kuo, “Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling Platforms,” in Proceedings of the 13th IEEE RTCSA, 2007.

[2] L. Sha, R. Rajkumar, and J. P. Lehoczky, “Priority Inheritance Protocols: an Approach to Real-Time Synchronization,” IEEE Transactions on Computers, vol. 39, no. 9, pp. 1175–1185, Sepember 1990.

[3] T. P. Baker, “A Stack-Based Resource Allocation Policy for Real-Time Processes,” in Proceedings of the 11th IEEE RTSS, December 4-7 1990, pp. 191-200.

[4] F. Zhang and S. T. Chanson, “Blocking-Aware Processor Voltage Scheduling for Real-Time Tasks,” ACM Transactions on Embedded Computing Systems, vol. 3, no. 2, pp. 307–335, 2004.

[5] J. Lee, K. Koh, and C.-g. Lee, “Multi-Speed DVS Algorithms for Periodic Tasks with Non-Preemptible Sections,” in Proceedings of the 13th IEEE RTCSA, 2007, pp. 459–468.

[6] A. M. Elewi, M. H. A. Awadalla, and M. I. Eladawy, “Energy-Efficient Multi-Speed Algorithm for Scheduling Dependent Real-Time Tasks,” in Proceedings of the ICCES, November 2008, pp. 237–242.

[7] R. Jejurikar and R. Gupta, “Energy Aware Task Scheduling with Task Synchronization for Embedded Real Time Systems,” IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 25, no. 6, pp. 1024–1037, 2006.

[8] Y.-W. Pan, “Energy-Efficient Task Synchronization for Real-Time Systems on Dynamic Voltage Scaling Platforms,” Master thesis, National Pingtung Institute of Commerce, Pingtung, Taiwan, 2009.

[9] J. Wu, Y.-W. Pan, and K.-L. Kao, “Energy-Efficient DVS Scheduling for Real-Time Tasks,” in Proceedings of the 2010 International Conference on Advanced Information Technologies (AIT 2010), April 23-24 2010, Taichung, Taiwan.

[10] K.-L. Kao, “DVS Scheduling of Real-Time Tasks with Abortable Critical Sections,” Master thesis, National Pingtung Institute of Commerce, Pingtung, Taiwan, 2011.

[11] J. Wu, “A Prediction-Based Approach for Energy-Efficient DVS Scheduling of Dependent Real-Time Tasks,” in Proceedings of the 17th IEEE RTAS, April 11-14 2011.

[12] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” Journal of the Association for Computing Machinery (JACM), vol. 20, no. 1, pp. 46–61, January 1973.

[13] M. Chen and K. Lin, “Dynamic Priority Ceilings: A Concurrency Control Protocol for Real-Time Systems,” Real Time Systems Journal, vol. 2, no. 1, pp. 325–346, 1990.

[14] Intel, Intel XScale Core Developer’s Manual, 2004.