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 previous work.

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

I. INTRODUCTION

In the past decades, many excellent approaches have been proposed so as to reduce energy consumption of real-time tasks on DVS platforms. A comprehensive survey of energy-efficient real-time scheduling can be found in [1]. Most works assume that tasks are independent, however, relatively little work has been 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 the stack resource policy (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 LeJurikar and Gupta [7] 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 processor which supports a set of \( k \) discrete speeds \( S = \{s_1, s_2, ..., s_k\} \), 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 \( E_C(t_1, t_2) \) 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 \( T = \{\tau_1, \tau_2, ..., \tau_n\} \) 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 period \( T_i \). Let \( \tau_{i,j} \) denote the \( j \)-th instance of task \( \tau_i \). The worst-case computation amount and the deadline of a task \( \tau_i \) are defined by \( C_i \) and \( D_i \). When \( \tau_i \) is executed at a speed \( s_{\tau_i} \), the worst-case execution time of \( \tau_i \) is \( C_i / s_{\tau_i} \). We consider well-formed tasks which satisfy \( 0 \leq C_i / s_{\tau_i} \leq D_i \) for all \( \tau_i \in T \). We also assume that the deadline is equal to the period, i.e., \( D_i = T_i \). The priority of a task \( \tau_i \) is defined by \( p_i \).

We assume that a set of \( m \) multiunit resources \( R = \{r_1, r_2, ..., r_m\} \) are accessed by tasks[3]. Each multiunit resource \( r_j \) has a fixed number of units, denoted as \( N_{r_j} \). A task \( \tau_i \) may make one or more requests for accessing multiunit resources during its execution. Let \( \mu(\tau_i) \) be the
number of units of resource \( r_j \) requested by task \( \tau_i \), 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 \[ Z_i = \langle z_{i,1}, z_{i,2}, ..., z_{i,n} \rangle \] be the list of \( \tau_i \)'s critical
sections. Each critical section \( z_{i,j} \) is a request which needs 
\( \mu(z_{i,j}) \) units of the resource \( R(z_{i,j}) \). 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 \( \tau_i \) is said to be
blocked by a lower-priority task \( \tau_j \)'s critical section if it has
to wait for \( r_j \) to exit the critical section in order to resume its
execution.

C. Problem Definitions

Let \( \text{lcm} \) denotes the least common multiple of all tasks' periods (also called hyperperiod). Since the taskset \( T \) repeats
an identical execution trace every hyperperiod, we only
need to examine the time interval \( (0, \text{lcm}) \) 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 \( T \) and a set
of shared multiunit resources \( R \). The problem is to schedule
\( T \) and to synchronize their accesses of shared multiunit
resources \( R \) on a non-ideal DVS processor such that

1. tasks have to meet their timing constraints, and
2. minimizes \( \text{lcm} \) \( PC(s(t)) \) dt . \( \square \)

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 the
same as those of the EDF and the SRP, respectively. Let \( \pi_i \) be the preemption level of a task \( \tau_i \). Under the BTS-SRP,
each task is assigned a fixed preemption level inversely
proportional to its deadline (i.e., \( \pi_i > \pi_j \iff D_i < D_j \)).

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

\[ CL_{r_j} = \max \{ 0 \cup \{ \pi_i : n_{r_j} < \mu_{r_j} (\tau_i) \} \} \]

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

\[ \pi_x = \max \{ 0, CL_{r_j}(n_{r_j}) \} \]

The rules for concurrency control of the BTS-SRP are as
the same as those of the SRP. Note that we use \( t_{i,j}^h \) to denote
the time that \( t_{i,j} \) 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.

\[ \sum_{k=1}^{n} \frac{C_k + B_k}{D_k} \leq s \]  

where \( B_k \) is the maximum blocking time of \( \tau_i \). Note that the
worst-case blocking time of \( \tau_i \) is \( B_i / s_x \) when the processor
speed is \( s_x \). The following equation shows the value of the
base processor speeds \( s \) which is the lowest processor speed
for executing tasks without violating their timing constraints.

\[ s = \max_{s \in \mathbb{R}} \{ s : \sum_{k=1}^{n} \frac{C_k + B_k}{D_k} \leq s \} \]  

C. Task Execution Speed and Blocking-Time Stealing

Firstly, all task's critical sections are assigned to be
executed at the base processor speed \( s \). It ensures that the
actual blocking time of any task \( \tau_i \) does not exceed \( B_i / s \).

Secondly, every task instance \( \tau_{i,j} \) is assigned to be executed
at speeds \( s_{i,j} \leq s \) 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 \( t_{i,j}^h \). 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 \( t_{i,j}^s \), then the blocking interval is \( [t_{i,j}^s, t_{i,j}^h] \).

Since all critical sections are executed at speeds \( s \), the
worst-case blocking time of \( \tau_{i,j} \) is \( B_i / s \), and \( t_{i,j}^s - t_{i,j}^h \leq B_i / s \).

For any task \( \tau_i \), its computation amount \( C_i \) consists of two
parts: \( c \) \( C_i \) and \( n \) \( C_i \), where \( c \) \( C_i \) is the total computation amount of its
critical sections, i.e., \( c \) \( C_i = \sum_{z_{i,j} \in Z_i} |z_{i,j}| \), and \( n \) \( C_i \) is the
computation amount of its non-critical part, i.e., \( n \) \( C_i = C_i - c \) \( C_i \).

When the actual blocking time is less than the worst-case
blocking time, we get additional \( B_i / s - (t_{i,j}^s - t_{i,j}^h) \) time
for executing the task instance \( \tau_{i,j} \). This additional
\( t_{i,j} \) is to slow down the execution speed from \( s \) 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 \( n \) \( C_i / s_{i,j} + c \) \( C_i / s \). For ensuring the
The primary performance metric of interest is the energy consumption of tasks, which is 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 \( P_C(s_i) = (0.08 + 1.52v_i^3) \text{ Watts} \) [14]. We set the supply voltage \( v_i = s_i \times 10 \), \( \forall s_i \in S \). The energy consumption in time interval \([t_1, t_2]\) can be obtained by \( \int_{t_1}^{t_2} P_C(s(t)) \text{ dt} \).

For generating feasible task sets, we set the utilization of tasks from 0.2 to 1 with an increment of 0.1. The period of a task was selected form 100 to 2000 by normal distribution. The worst-case computation amount of task was selected from 10 to 300 by normal distribution. The number of shared multiunit resources is 5 to 10 and the number of units for each resource is 1 to 5.

The schedulability of all tasks, \( \tau_{ij} \)'s execution time cannot exceed \( C_i/s^b + B_i/s^b - (t^b_{ij} - t^b_{ij}) \). Hence, the processor speed \( s^b_{ij} \) can be calculated as follows:

\[
s^b_{ij} = \max_{s_k \in S} \{s_k: \frac{s^b_{nC_i}}{B_i - s^b(t^b_{ij} - t^b_{ij})} + nC_i \leq s_k \} \tag{3}
\]

For each task \( \tau_i \), 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 at \( s_{max} \)). 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 \( s^b \) 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 except 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**


