Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 1237438, 7 pages
https://doi.org/10.1155/2017/1237438

Research Article

Best Speed Fit EDF Scheduling for Performance Asymmetric Multiprocessors

Peng Wu[1] and Minsoo Ryu[2]

[1] *Department of Electronics and Computer Engineering, Hanyang University, Seoul, Republic of Korea*
[2] *Department of Computer Science and Engineering, Hanyang University, Seoul, Republic of Korea*

Correspondence should be addressed to Minsoo Ryu; msryu@hanyang.ac.kr

Received 10 October 2016; Revised 19 January 2017; Accepted 1 February 2017; Published 20 February 2017

Academic Editor: Alessandro Gasparetto

In order to improve the performance of a real-time system, asymmetric multiprocessors have been proposed. The benefits of improved system performance and reduced power consumption from such architectures cannot be fully exploited unless suitable task scheduling and task allocation approaches are implemented at the operating system level. Unfortunately, most of the previous research on scheduling algorithms for performance asymmetric multiprocessors is focused on task priority assignment. They simply assign the highest priority task to the fastest processor. In this paper, we propose BSF-EDF (best speed fit for earliest deadline first) for performance asymmetric multiprocessor scheduling. This approach chooses a suitable processor rather than the fastest one, when allocating tasks. With this proposed BSF-EDF scheduling, we also derive an effective schedulability test.

1. Introduction

Modern semiconductor technology adopts the use of multiple processors in their motherboard design because of the growing demand of performance and saving energy which cannot be handled by single processor systems. In this regard, performance asymmetric multiprocessors, where individual cores possess different performance, are believed to provide improved performance and low power consumption when compared to performance symmetric multiprocessors, where the cores are identical [1]. This is because such heterogeneous architecture allows the allocation of computing resources according to the needs of an application and better handling of dynamic workload requirements by exploiting speed of the fast cores and power efficiency of the slow cores available. One example of heterogeneous multiprocessor architectures is the ARM big.LITTLE architectures that combine slow and fast processing cores that possess identical instruction set but different execution speeds. The ARM big.LITTLE architectures are already used by Samsung in their Galaxy Note 5 and S6 Edge.

For embedded real-time systems, in addition to resource and power optimization, guaranteeing the deadlines of a real-time system is a critical requirement. Scheduling on such a performance asymmetric multicore platform is much more challenging than scheduling on identical multicore platform since the processing speed depends not only on the processor type, but also on the task executed. Thus, on heterogeneous multicore platforms, a decision that which type of processor will execute a given task over a period of time is also important.

Liu and Layland proposed earliest deadline first (EDF) scheduling algorithm on real-time single processor first in 1973 [2]. It is proved to be an optimal algorithm on single processor. Baker demonstrated an efficiently computable schedulability test for EDF scheduling on an asymmetric multiprocessors platform [3]. However, most of the previous research on EDF scheduling algorithm and relative scheduling algorithms for performance asymmetric multiprocessors is focused on the task priority approach [4–8]. They simply assign the highest priority task to the fastest processor. Since processors have different processing speeds in performance asymmetric multiprocessors, we believe that task allocation to an appropriate processor is also an important issue to be considered.

In the current research we focus on providing an improved scheduling algorithm based on EDF for handling real-time tasks for performance asymmetric multiprocessor systems. More specifically, we propose a new EDF scheduling algorithm, best speed fit EDF (BSF-EDF). For BSF-EDF, the priorities of the tasks are chosen according to the original EDF scheduling algorithm, whereas it differs in terms of task allocation to processors, such that in the former approach the highest priority task is assigned to a faster processor and in the latter approach the highest priority task is allocated to the appropriate idle processor. We also obtain a new schedulability test for the BSF-EDF on performance asymmetric multiprocessors.

The rest of the paper is organized as follows. In Section 2, we describe system model and background. Section 3 presents the BSF-EDF scheduling with an example. In Section 4, we describe the proposed schedulability test for the BSF-EDF scheduling algorithm.

2. System Model and Background

We consider an m-core performance asymmetric multiprocessor system and we use π to denote the set of m performance asymmetric processors. We use s_i to represent the processing speed of any processor P_i. For convenience we use 1 to represent the speed of the slowest processor, such that the speed of faster processors can be represented by a multiple of 1; for example, $s_i = 3$. We also assume that the speeds of processors are specified in nondecreasing order: $s_i \le s_{i+1}$ for all i. Let S_i be the cumulative sum of processor speeds of the first i processors in a performance asymmetric multiprocessors platform π:

$$S_i = \sum_{j=1}^{i} s_j. \tag{1}$$

Immediately, S_m denotes the cumulative sum of π.

In this paper, we consider a sporadic task model. Let $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_n\}$ be a set of n sporadic tasks. A sporadic task τ_i is defined by a 3-tuple (e_i, d_i, T_i), where e_i is the worst case execution time when it is executed in the lowest speed processor, d_i is the relative constrained deadline such that $e_i \le d_i \le T_i$, and T_i is the minimum interarrival time, which is also referred as period of the task. The minimum interarrival time T_i means that any two consecutive requests of τ_i are separated by at least T_i. We use $\tau_{i,j}$ to represent the jth job of task τ_i and $A_{i,j}$ to denote the arrival time of $\tau_{i,j}$. Note that a task can have different execution time based on different speeds of processors. Therefore, a job executing on the jth processor for a duration of t time can receive $t \times s_j$ units of execution.

Now we introduce several further notions that we will use for schedulability analysis. Since processors have varying speeds, we use the slowest processor to measure the worst case execution time e_i of a task τ_i. The utilization u_i of a task τ_i is the ratio of the worst case execution time to its period, that is, e_i/T_i. The total utilization $U_{\text{sum}}(\Gamma)$ and the largest task utilization $u_{\max}(\Gamma)$ of a task set are defined as follows:

$$U_{\text{sum}}(\Gamma) = \sum_{\tau_i \in \Gamma} u_i, \qquad u_{\max}(\Gamma) = \max_{\tau_i \in \Gamma} u_i. \tag{2}$$

The density δ_i of a task τ_i is defined to be the ratio of e_i/d_i, that is, ratio of the worst case execution time to the relative deadline. The maximum density $\delta_{\max}(\Gamma)$ of a task set Γ is defined as follows:

$$\delta_{\max}(\Gamma) = \max_{\tau_i \in \Gamma} \delta_i. \tag{3}$$

The concepts of processor demand bound function $\text{DBF}(\tau_i, \Delta t)$ and processor load are used in the analysis of multiprocessor scheduling and are well-studied in detail in [9]. $\text{DBF}(\tau_i, \Delta t)$ is used to represent an upper bound on the maximum cumulative processing time demand by jobs of τ_i that arrive in and also have deadline within any interval of length Δt. It has been shown that

$$\text{DBF}(\tau_i, \Delta t) \overset{\text{def}}{=} \begin{cases} \max\left(0, \left(\left\lfloor \dfrac{\Delta t - d_i}{T_i} \right\rfloor + 1\right) e_i\right) & \text{if } \Delta t \ge d_i \\ e_i & \text{if } \Delta t < d_i. \end{cases} \tag{4}$$

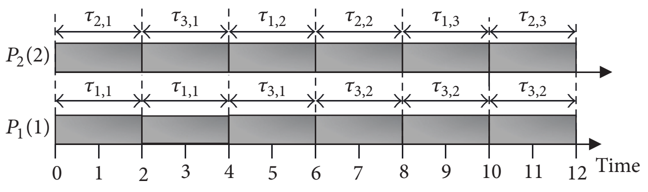
The processor load parameter is based on the DBF function which is the maximum value of the ratio of processor demand bound and length of the time interval Δt [10].

$$\text{LOAD}(\Gamma) \overset{\text{def}}{=} \max_{\Delta t > 0} \left(\frac{\sum_{\tau_i \in \Gamma} \text{DBF}(\tau_i, \Delta t)}{\Delta t} \right). \tag{5}$$

3. BSF-EDF Scheduling Algorithm for Performance Asymmetric Multiprocessor Platform

Our scheduling algorithm proposed in this paper is a priority driven scheduling algorithm, which means that the highest priority task always runs first. Moreover, we choose global multiprocessor scheduling policy. Global scheduling maintains a single system-wide queue of ready tasks, from which tasks are extracted at run-time, to be scheduled on the available computing resources. Global scheduling is more appropriate for open systems, as there is no need to run load balancing algorithms when the set of tasks changes. We also allow preemption of tasks in our proposed BSF-EDF scheduling algorithm and migration of tasks between different speed processors is also permitted.

Now there are two crucial issues to be addressed, task priority assignment and task allocation to processors. For BSF-EDF scheduling we assign the priorities to tasks according to their absolute deadline, such that the task with earlier deadline has higher priority. In previous research work, individuals simply assign the highest priority task to the fastest idle processor. Contrary to this, we assign the highest priority task to the appropriate idle processor rather than the fastest idle one. The appropriate processor is the slowest speed processor which ensures that a task can be executed without missing its deadline. In other words, we choose the slowest speed idle processor whose speed is no less than the utilization of the task. If there does not exist such appropriate processor, we allocate the task to the slowest idle processor. We should reschedule all the tasks whenever a schedule event



$\tau_{2,1}$ $\tau_{3,1}$ $\tau_{1,2}$ $\tau_{2,2}$ $\tau_{1,3}$ $\tau_{2,3}$
$P_2(2)$
$\tau_{1,1}$ $\tau_{1,1}$ $\tau_{3,1}$ $\tau_{3,2}$ $\tau_{3,2}$ $\tau_{3,2}$
$P_1(1)$
0 1 2 3 4 5 6 7 8 9 10 11 12 Time

FIGURE 1: Task schedule under BSF-EDF.

occurs. Schedule events include two cases, when (a) a running job is finished and (b) a new job has arrived. Specifically, we say that (1) no processor is idle while there is an active task, (2) a scheduled task always executes on the best speed fit processor, if there exists such a processor, and (3) higher priority task always runs first.

Example 1. Figure 1 manifests a simple BSF-EDF scheduling example on a performance asymmetric multiprocessor platform. Consider a performance asymmetric multiprocessor platform $\pi = \{1, 2\}$ which has two processors and a task set $\Gamma = \{\tau_1(4,4,4), \tau_2(4,4,4), \tau_3(6,6,6)\}$. Initially, $\tau_{1,1}$ has the highest priority, and P_1 is the appropriate idle processor, so we allocate it to P_1. The second priority job $\tau_{2,1}$ is allocated to processor P_2. Job $\tau_{2,1}$ finishes at time $t = 2$; therefore, time $t = 2$ is a schedule event time. All tasks should be rescheduled. $\tau_{1,1}$ still has the highest priority; we allocate it to P_1, and we allocate second priority job $\tau_{3,1}$ to P_2. $\tau_{1,1}$ finishes its execution at time $t = 4$, so a schedule event appears. Now $\tau_{3,1}$ has the highest priority, so we allocate it to P_1, and we allocate job $\tau_{1,2}$ to P_2. It is easy to see that all tasks will satisfy their deadline under BSF-EDF. The processors are used to their 100% utilization. As shown in the figure, $t = \{2, 4, 6, 8, 10, 12\}$ are schedule event points.

4. BSF-EDF Schedulability Analysis for Performance Asymmetric Multiprocessor Platform

With respect to a given platform, a given sporadic task set is said to be BSF-EDF schedulable if the schedule is able to meet all deadlines for every collection of jobs that may be generated by the task set. In this section, we derive a schedulability test for the BSF-EDF scheduling on performance asymmetric multiprocessors.

We first obtain a necessary condition when a task misses its deadline for the BSF-EDF scheduling. By taking the contrapositive of the condition, we obtain a sufficient schedulability condition.

Suppose that a job $\tau_{i,j}$ of task τ_i is the one to first miss its deadline at time-instant $A_{i,j} + d_i$ (see Figure 2). $A_{i,j}$ denotes the arrival time of job $\tau_{i,j}$. Due to the properties of BSF-EDF scheduling, we discard the legal sequence of jobs with deadlines later than $A_{i,j} + d_i$. We only consider the BSF-EDF schedule of the remaining legal sequence of job requests, since the later deadline jobs have no effect on the scheduling of earlier deadline ones under BSF-EDF. It follows that a

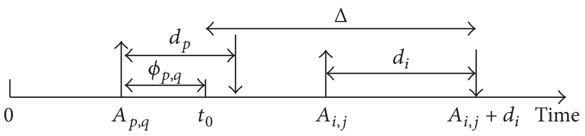


FIGURE 2: A job $\tau_{i,j}$ of task τ_i arrives at $A_{i,j}$ and misses its deadline at time-instant $A_{i,j} + d_i$.

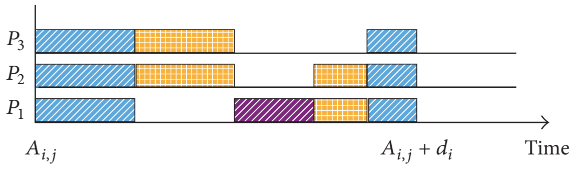


FIGURE 3: An example: there are three processors, and the total length of blue color units is B_3. The total length of orange color units is B_2. The total length of purple color units is B_1.

deadline miss of τ_i occurs at time-instant $A_{i,j} + d_i$ (and this is the earliest deadline miss).

For a given task set $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_n\}$, let $W(t_p, t_q)$ be the total amount of work done during $[t_p, t_q)$, and let $R(t_p, t_q)$ be the total amount of processor time demand in $[t_p, t_q)$ required by all the jobs in this legal sequence of task set Γ.

We derive a necessary condition for a task τ_i to miss its deadline, by following three steps: (1) deriving a lower bound on the total amount of processor time demand during the time interval $[t_0, A_{i,j} + d_i)$ in Section 4.1, (2) deriving an upper bound on the total amount of processor time demand during the time interval $[t_0, A_{i,j} + d_i)$ in Section 4.2, and (3) combining the lower bound and upper bound into a necessary condition for task τ_i missing its deadline. Note that t_0 is a special time point that will be explained in the coming section.

4.1. Lower Bound. In this section, we obtain a lower bound of $R(t_0, A_{i,j} + d_i)$.

Consider job $\tau_{i,j}$ arriving at time $A_{i,j}$ misses its deadline at time-instant $A_{i,j} + d_i$. Let B_v be the total duration over $[A_{i,j}, A_{i,j} + d_i)$ for which exactly v processors are busy in this BFS-EDF schedule, $0 \le v \le m$. Note that B_0 is necessarily zero, since $\tau_{i,j}$ does not finish before its deadline. Figure 3 manifests an example to understand this definition. S'_v denotes the cumulative speed of the currently running v processors. We can then write

$$W\left(A_{i,j}, A_{i,j} + d_i\right) = \sum_{v=1}^{m} S'_v B_v. \tag{6}$$

Since $\sum_{v=1}^{m} S'_v B_v = S_m B_m + \sum_{v=1}^{m-1} S'_v B_v$ and $S_m B_m = S_m (d_i - \sum_{v=1}^{m-1} B_v)$, we can write

$$W\left(A_{i,j}, A_{i,j} + d_i\right) = S_m d_i - \sum_{v=1}^{m-1} \left(S_m B_v - S'_v B_v\right). \tag{7}$$

As we mentioned in Section 2, S_v means the cumulative speed of the first v lowest speed processors. The tasks may be allocated to the faster speed processors under BSF-EDF; therefore we can get

$$S'_v \geq S_v. \quad (8)$$

From (7) and (8) above, we conclude that

$$\begin{aligned} W\left(A_{i,j}, A_{i,j} + d_i\right) &\geq S_m d_i - \sum_{v=1}^{m-1} \left(S_m B_v - S_v B_v\right) \\ &= S_m d_i - \sum_{v=1}^{m-1} \frac{S_m - S_v}{s_1} s_1 B_v. \end{aligned} \quad (9)$$

Let $\lambda(\pi) = \max_{1 \leq i \leq m}(\sum_{j=i+1}^{m} s_j / s_1)$. It immediately follows that

$$\lambda(\pi) \geq \frac{S_m - S_v}{s_1} \quad \text{for } 1 \leq v \leq m-1. \quad (10)$$

Combining (9) and (10), we have

$$W\left(A_{i,j}, A_{i,j} + d_i\right) \geq S_m d_i - \lambda(\pi) \sum_{v=1}^{m-1} s_1 B_v. \quad (11)$$

Due to the work conserving property of BSF-EDF, $\tau_{i,j}$ must always be running in one of processors in time interval $[A_{i,j}, A_{i,j} + d_i)$, except when all the processors are busy. The processor time that $\tau_{i,j}$ receives during $[A_{i,j}, A_{i,j} + d_i)$ cannot be less than $\sum_{v=1}^{m-1} s_1 B_v$, which is the worst case processor time when $\tau_{i,j}$ is running on the slowest processor s_1. The processor time that $\tau_{i,j}$ receives must not exceed e_i, since job $\tau_{i,j}$ misses its deadline. Thus, we have

$$e_i > \sum_{v=1}^{m-1} s_1 B_v. \quad (12)$$

By applying this to (11), we have

$$\begin{aligned} W\left(A_{i,j}, A_{i,j} + d_i\right) &> S_m d_i - \lambda(\pi) e_i, \\ \frac{W\left(A_{i,j}, A_{i,j} + d_i\right)}{d_i} &> S_m - \lambda(\pi) \delta_i. \end{aligned} \quad (13)$$

Since $\delta_{\max}(\Gamma) \geq \delta_i$, we can write

$$\frac{W\left(A_{i,j}, A_{i,j} + d_i\right)}{d_i} > S_m - \lambda(\pi) \delta_{\max}(\Gamma). \quad (14)$$

Let

$$\mu = S_m - \lambda(\pi) \delta_{\max}(\Gamma). \quad (15)$$

Then we have

$$W\left(A_{i,j}, A_{i,j} + d_i\right) > \mu d_i. \quad (16)$$

Based on this, we now obtain a lower bound of $R(t_0, A_{i,j} + d_i)$, where t_0 is the earliest time point t such that $t \leq A_{i,j}$ and $W(t, A_{i,j} + d_i) > \mu(A_{i,j} + d_i - t)$. Note that we can always find such a time instant since at least $A_{i,j}$ satisfied $A_{i,j} \leq A_{i,j}$ and $W(A_{i,j}, A_{i,j} + d_i) > \mu(A_{i,j} + d_i - A_{i,j})$. Let $\Delta = A_{i,j} + d_i - t_0$; then we finally have

$$W\left(t_0, A_{i,j} + d_i\right) > \mu\Delta. \quad (17)$$

Note that the job $\tau_{i,j}$ cannot be completed before time $A_{i,j} + d_i$, so the total amount of work processed can never exceed the amount of processor time demand. Thus, we have

$$R\left(t_0, A_{i,j} + d_i\right) \geq W\left(t_0, A_{i,j} + d_i\right). \quad (18)$$

It immediately follows that

$$R\left(t_0, A_{i,j} + d_i\right) > \mu\Delta. \quad (19)$$

As a result, $\mu\Delta$ is a lower bound of the total amount of processor time demand during the time interval $[t_0, A_{i,j} + d_i)$.

4.2. Upper Bound. We now obtain an upper bound of $R(t_0, A_{i,j} + d_i)$ of the total amount of processor time demand. For this, we need to consider two types of jobs, regular jobs that arrived at or after t_0 and the jobs that arrive before t_0 but have not completed execution in the BSF-EDF schedule by the time-instant t_0. We will refer to jobs arriving prior to t_0 that need execution over $[t_0, A_{i,j} + d_i)$ as carry-in jobs. We use R_c to denote the total amount of processor time demand caused by carry-in jobs and use R_r to denote the total amount of processor time demand caused by the regular jobs.

We first compute the upper bound of R_c by (1) determining an upper bound on the remaining processor time demand for each carry-in job, (2) determining a maximum value of the number of carry-in jobs, and (3) combining the per-job bounds to obtain an upper bound on the total remaining processor time demand of all carry-in jobs.

Lemma 2. *Each carry-in job has less than $\delta_{\max}(\Gamma)\Delta$ remaining execution requirement at time-instant t_0.*

Proof. Let us consider a carry-in job $\tau_{p,q}$ of a task τ_p that arrives at time-instant $A_{p,q} < t_0$ and has not completed its execution by time t_0 (see Figure 2). Let $\phi_{p,q} = t_0 - A_{p,q}$. Let $e'_{p,q}$ be the processing time that $\tau_{p,q}$ receives during time interval $[A_{p,q}, t_0)$. By using the same approach as in Section 4.1, let I_v be the total duration over $[A_{p,q}, t_0)$ for which exactly v processors are busy in this BSF-EDF schedule. J_v denotes the cumulative speed of the currently running v processors over $[A_{p,q}, t_0)$. Thus, the total amount of work processed in $[A_{p,q}, t_0)$ can be calculated by $S_m \phi_{p,q} - \sum_{v=1}^{m-1}(S_m I_v - J_v I_v)$. Since $J_v \geq S_v$, it immediately follows that

$$e'_{p,q} \geq S_m \phi_{p,q} - \sum_{v=1}^{m-1} \left(S_m I_v - S_v I_v\right). \quad (20)$$

Since we can also express the total amount of work done during the time interval $[A_{p,q}, t_0)$ by $W(A_{p,q}, A_{i,j} + d_i) - W(t_0, A_{i,j} + d_i)$, we can write

$$W\left(A_{p,q}, A_{i,j} + d_i\right) - W\left(t_0, A_{i,j} + d_i\right) = e'_{p,q}. \quad (21)$$

By applying (20) to (21), we have

$$\begin{aligned} &W\left(A_{p,q}, A_{i,j}+d_i\right) - W\left(t_0, A_{i,j}+d_i\right) \\ &\quad \geq S_m\phi_{p,q} - \sum_{v=1}^{m-1}\left(S_mI_v - S_vI_v\right). \end{aligned} \tag{22}$$

By the definition of t_0 (see (17)), it must satisfy

$$W\left(t_0, A_{i,j}+d_i\right) > \mu\Delta. \tag{23}$$

Note that t_0 is the earliest time point t such that $t \leq A_{i,j}$ and $W(t, A_{i,j}+d_i) > \mu(A_{i,j}+d_i-t)$, and we can also get

$$W\left(A_{p,q}, A_{i,j}+d_i\right) \leq \mu\left(\Delta + \phi_{p,q}\right). \tag{24}$$

It immediately follows that

$$\begin{aligned} \mu\left(\Delta+\phi_{p,q}\right) - \mu\Delta > W&\left(A_{p,q}, A_{i,j}+d_i\right) \\ &- W\left(t_0, A_{i,j}+d_i\right). \end{aligned} \tag{25}$$

By combining it to (22), we have

$$\begin{aligned} \mu\left(\Delta+\phi_{p,q}\right) - \mu\Delta > S_m\phi_{p,q} - \sum_{v=1}^{m-1}\left(S_mI_v - S_vI_v\right) &\Longrightarrow \\ \mu\phi_{p,q} > S_m\phi_{p,q} - \sum_{v=1}^{m-1}\left(S_mI_v - S_vI_v\right) &\Longrightarrow \\ \mu\phi_{p,q} > S_m\phi_{p,q} - \sum_{v=1}^{m-1}\frac{S_m - S_v}{s_1}s_1I_v&. \end{aligned} \tag{26}$$

Since

$$\begin{aligned} \lambda(\pi) &= \max_{1\leq i\leq m}\frac{\sum_{j=i+1}^{m} s_j}{s_1}, \\ \lambda(\pi) &\geq \frac{S_m - S_v}{s_1} \quad \text{for } 1 \leq v \leq m-1, \end{aligned} \tag{27}$$

we can write

$$S_m\phi_{p,q} - \sum_{v=1}^{m-1}\frac{S_m - S_v}{s_1}s_1I_v \geq S_m\phi_{p,q} - \lambda(\pi)\sum_{v=1}^{m-1}s_1I_v. \tag{28}$$

Equation (26) then becomes

$$\mu\phi_{p,q} > S_m\phi_{p,q} - \lambda(\pi)\sum_{v=1}^{m-1}s_1I_v. \tag{29}$$

Observe that the amount of execution that carry-in job $\tau_{p,q}$ receives over $[A_{p,q}, t_0)$ is no less than $\sum_{v=1}^{m-1} I_v s_1$ since $\tau_{p,q}$ must be executing on one of the processors during any instant, when any processor is idle. Thus, we have

$$e'_{p,q} \geq \sum_{v=1}^{m-1}s_1I_v. \tag{30}$$

It immediately follows that

$$S_m\phi_{p,q} - \lambda(\pi)\sum_{v=1}^{m-1}s_1I_v \geq S_m\phi_{p,q} - \lambda(\pi)e'_{p,q}. \tag{31}$$

By (29) and (31), we can write

$$\mu\phi_{p,q} > S_m\phi_{p,q} - \lambda(\pi)e'_{p,q}. \tag{32}$$

Using $\mu = S_m - \lambda(\pi)\delta_{\max}(\Gamma)$ (defined in Section 4.1) in (32), it follows that

$$\left(S_m - \lambda(\pi)\delta_{\max}(\Gamma)\right)\phi_{p,q} > S_m\phi_{p,q} - \lambda(\pi)e'_{p,q}, \tag{33}$$

resulting in

$$e'_{p,q} > \delta_{\max}(\Gamma)\phi_{p,q}. \tag{34}$$

Note that $e_{p,q} = d_p\delta_p$ and that the remaining processing time demand of the carry-in job $\tau_{p,q}$ of task is $e_{p,q} - e'_{p,q}$. Thus, we have

$$e_{p,q} - e'_{p,q} < d_p\delta_p - \delta_{\max}(\Gamma)\phi_{p,q}. \tag{35}$$

The absolute deadline of carry-in job $\tau_{p,q}$ is no later than time $A_{i,j}+d_i$; that is, $d_p - \phi_{p,q} \leq \Delta$. Also, $\delta_p \leq \delta_{\max}(\Gamma)$; therefore

$$\begin{aligned} e_{p,q} - e'_{p,q} &< d_p\delta_p - \delta_{\max}(\Gamma)\phi_{p,q}, \\ e_{p,q} - e'_{p,q} &< d_p\delta_{\max}(\Gamma) - \delta_{\max}(\Gamma)\phi_{p,q}, \\ e_{p,q} - e'_{p,q} &< \delta_{\max}(\Gamma)\left(d_p - \phi_{p,q}\right), \\ e_{p,q} - e'_{p,q} &< \delta_{\max}(\Gamma)\Delta. \end{aligned} \tag{36}$$

□

As a result, an upper bound of the remaining processor time demand for each carry-in job can be expressed as $\delta_{\max}(\Gamma)\Delta$.

We obtained the upper bound on the remaining processor time demand for each carry-in job. We will now determine the maximum value of the number of carry-in jobs.

Lemma 3. *The number of carry-in jobs is bounded by*

$$\Omega = \max\left\{\omega : S_\omega < \mu\right\}. \tag{37}$$

Proof. Let ε be an arbitrarily small positive number. From the definition of t_0, it follows that

$$\begin{aligned} W\left(t_0, A_{i,j}+d_i\right) &> \mu\Delta, \\ W\left(t_0 - \varepsilon, A_{i,j}+d_i\right) &\leq \mu(\varepsilon + \Delta). \end{aligned} \tag{38}$$

Therefore, the work processed during $[t_0 - \varepsilon, t_0)$ is less than $\mu\varepsilon$ since $W(t_0 - \varepsilon, A_{i,j}+d_i) - W(t_0, A_{i,j}+d_i) < \mu\varepsilon$. Based on the definition of $\mu \leq S_m$, it follows that the work processed during $[t_0 - \varepsilon, t_0)$ is less than εS_m. Therefore some processor is idle during $[t_0 - \varepsilon, t_0)$. This implies that all carry-in jobs are running in this interval and the number of carry-in jobs is the same as the number of busy processors. Let ω be the number of busy processors over $[t_0 - \varepsilon, t_0)$ and let Ω be an upper bound of ω. Thus, we have

$$\Omega = \max \{\omega : S_\omega < \mu\}. \tag{39}$$

□

Therefore the upper bound of R_c is $\Omega\Delta\delta_{\max}(\Gamma)$.

Now let us consider the upper bound of the total amount of processor time demand caused by regular jobs. Fortunately, it is a well-studied subject, and algorithms are known for computing DBF and LOAD [9, 10]. Baruah et al. showed that the upper bound of R_r can be calculated by $\Delta \times \mathrm{LOAD}(\Gamma)$, where

$$\begin{aligned} &\mathrm{LOAD}(\Gamma) \overset{\text{def}}{=} \max_{\Delta t>0} \left(\frac{\sum_{\tau_i \epsilon \Gamma} \mathrm{DBF}(\tau_i, \Delta t)}{\Delta t} \right), \\ &\mathrm{DBF}(\tau_i, \Delta t) \\ &\quad \overset{\text{def}}{=} \begin{cases} \max\left(0, \left(\left\lfloor \dfrac{\Delta t - d_i}{T_i} \right\rfloor + 1\right) e_i\right) & \text{if } \Delta t \geq d_i \\ e_i & \text{if } \Delta t < d_i. \end{cases} \end{aligned} \tag{40}$$

As a result, an upper bound of the total amount of processor time demand can be expressed as

$$R(t_0, A_{i,j} + d_i) \leq \mathrm{LOAD}(\Gamma)\Delta + \Omega\Delta\delta_{\max}(\Gamma). \tag{41}$$

4.3. BSF-EDF Schedulability Test. We now describe a sufficient schedulability test for the BSF-EDF scheduling algorithm.

Theorem 4. *A task set $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_n\}$ is schedulable by BSF-EDF on an m-core performance asymmetric multiprocessor platform $\pi = \{P_1, P_2, \ldots, P_m\}$, with processing speeds $s_1 = s_2 = \cdots = s_q = 1$, and $s_i > 1$ for $q < i \leq m$, if it satisfies the following condition:*

$$\mathit{LOAD}(\Gamma) \leq \mu - \Omega\delta_{\max}(\Gamma), \tag{42}$$

where μ and Ω are defined in (15) and (37), respectively.

Proof. We prove it through the principle of contradiction. We obtain necessary condition for the job $\tau_{i,j}$ missing its deadline. Negating this condition yields a sufficient condition for the BSF-EDF schedulability.

From the foregoing discussion, we already got the upper bound and lower bound of the total amount of processor time demand during the time interval $[t_0, A_{i,j} + d_i)$. As a result, we obtain the following bound on $R(t_0, A_{i,j} + d_i)$:

$$\begin{aligned} &R(t_0, A_{i,j} + d_i) \leq \mathrm{LOAD}(\Gamma)\Delta + \Omega\Delta\delta_{\max}(\Gamma), \\ &R(t_0, A_{i,j} + d_i) > \mu\Delta. \end{aligned} \tag{43}$$

Based on (43), we can get the necessary condition for a task τ_i to miss its deadline:

$$\begin{aligned} &\mathrm{LOAD}(\Gamma)\Delta + \Omega\Delta\delta_{\max}(\Gamma) > \mu\Delta \\ &\quad \equiv \mathrm{LOAD}(\Gamma) + \Omega\delta_{\max}(\Gamma) > \mu \\ &\quad \equiv \mathrm{LOAD}(\Gamma) > \mu - \Omega\delta_{\max}(\Gamma). \end{aligned} \tag{44}$$

Equivalently, the negation of this condition is sufficient to ensure BSF-EDF schedulability:

$$\mathrm{LOAD}(\Gamma) \leq \mu - \Omega\delta_{\max}(\Gamma), \tag{45}$$

which is as claimed in Theorem 4. □

5. Conclusions

In this paper, we presented a unique approach to task allocation in a performance asymmetric multiprocessor system based on EDF, called BSF-EDF. In BSF-EDF we chose the appropriate slowest speed processor to execute the highest priority task. To our knowledge, this is the first piece of work that considers the task allocation problem on performance asymmetric multiprocessors. We hold the view that choosing an appropriate processor instead of a fastest one gives opportunities of improved scheduling, by reserving fast processors for future tasks that will arrive later. We have also presented an effective schedulability test for performance asymmetric multiprocessors based on the properties of the proposed BSF-EDF.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported partly by the ICT R&D Program of MSIP/IITP [R01141600460001002, Software Black Box for Highly Dependable Computing] and partly by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (NRF-2015R1A5A7037751).

References

[1] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero, and E. Ayguadé, "Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors," *IEEE Computer Architecture Letters*, vol. 5, no. 1, pp. 14–17, 2006.

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, pp. 46–61, 1973.

[3] T. P. Baker, "Multiprocessor EDF and deadline monotonic schedulability analysis," in *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS '03)*, pp. 120–129, Washington, DC, USA, December 2003.

[4] J. Lee and I. Shin, "EDZL schedulability analysis in real-time multicore scheduling," *IEEE Transactions on Software Engineering*, vol. 39, no. 7, pp. 910–916, 2013.

[5] M. Cirinei and T. P. Baker, "EDZL scheduling analysis," in *Proceedings of the 19th Euromicro Conference on Real-Time Systems (ECRTS '07)*, pp. 9–18, Pisa, Italy, July 2007.

[6] H. W. Wei, Y. H. Chao, S. S. Lin, K. J. Lin, and W. K. Shih, "Current results on EDZL scheduling for multiprocessor real-time systems," in *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '07)*, pp. 120–130, 2007.

[7] Y.-H. Chao, S.-S. Lin, and K.-J. Lin, "Schedulability issues for EDZL scheduling on real-time multiprocessor systems," *Information Processing Letters*, vol. 107, no. 5, pp. 158–164, 2008.

[8] P. Wu and M. Ryu, "EDZL scheduling and schedulability analysis for performance asymmetric multiprocessors," *International Journal of Foundations of Computer Science*, vol. 27, no. 1, pp. 1–14, 2016.

[9] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proceedings of the 11th Real-Time Systems Symposium (RTSS '90)*, pp. 182–190, IEEE, Lake Buena Vista, Fla, USA, December 1990.

[10] T. P. Baker, N. Fisher, and S. Baruah, "Algorithms for determining the load of a sporadic task system," Tech. Rep. TR-051201, 2005.