applied sciences

MDPI

Article

Three Processor Allocation Approaches towards EDF Scheduling for Performance Asymmetric Multiprocessors

Peng Wu [1,2,*], Zhi Li [1,2], Tao Yan [1,2] and Yingchun Li [3]

[1] School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China
[2] Institute of Big Data Science and Industry, Shanxi University, Taiyuan 030006, China
[3] School of Information Science and Engineering, Harbin Institute of Technology, Weihai 264209, China
* Correspondence: pengwu@sxu.edu.cn

Abstract: With the rapid development of high-performance computing and parallel computing technology, by virtue of its cost-effectiveness, strong scalability, and easy programming, the multiprocessor system has gradually become the mainstream computing platform. Meanwhile, a growing number of researchers pay attention to the performance of multiprocessor systems, especially the task scheduling problem, which has an important impact on the system performance. Most of the current research works on task scheduling algorithms are based on the homogeneous computing environment. On the contrary, research works focusing on more complex performance asymmetric multiprocessor environments still remain rare. In this paper, we compare the effects of three earliest deadline first algorithms under different processor allocation strategies on performance asymmetric multiprocessors. We propose an efficient schedulability analysis for an allocation strategy that assigns high-priority tasks to the slowest idle processor. Experimental results show that the strategy of allocating processors with optimum speeds for high-priority tasks can schedule more task sets than the other two allocation strategies. The strategy that prioritizes the slowest processors for high-priority tasks has the smallest number of task migrations, and the strategy has the highest effective processor utilization.

Keywords: EDF scheduling algorithm; performance asymmetric multiprocessors; schedulability analysis

check for updates

Citation: Wu, P.; Li, Z.; Yan, T.; Li, Y. Three Processor Allocation Approaches towards EDF Scheduling for Performance Asymmetric Multiprocessors. *Appl. Sci.* **2023**, *13*, 5318. https://doi.org/10.3390/app13095318

Academic Editors: Jie Lin and Arkadiusz Gola

Received: 11 February 2023
Revised: 4 April 2023
Accepted: 21 April 2023
Published: 24 April 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the continuous increase in people's requirements for computing power, traditional single-processor systems cannot satisfy the needs of users. Real-time systems are applied on performance asymmetric multiprocessor platforms to meet the high computing needs of users effectively [1]. A performance asymmetric multiprocessor architecture allows the system to allocate computing resources according to demand and make full use of available fast and slow cores to handle dynamic workload demands; hence, it can improve the performance and reduce the power consumption [2]. Moreover, it has shown strong performance advantages in some specific applications, such as smartphones, high-end digital cameras, and other electronic devices [3]. All these high-level products can boost our lives in different aspects. This has also led to the emergence of services such as the Internet of Things, intelligent transportation, and health monitoring, and people's high computing needs continue to increase. Researchers have conducted a lot of research on this. Irtija et al. proposed an efficient edge computing solution to meet the computing needs of Internet of Things nodes [4]. Experiments show that the number of satisfied users increases by 40% with only a 1% drop in maximum accuracy.

Ensuring the real-time performance of tasks is the most important goal in real-time systems. For real-time systems, optimizing resources and power consumption is important. However, the most important aspect is to ensure the real-time nature of the task. However, current research is still unable to guarantee real-time requirements. This means that the

performance asymmetric multiprocessors cannot make full use of the performance in the real-time system [5].

To track this challenge, researchers have proposed some works. The earliest deadline first (EDF) scheduling algorithm was proposed by Liu et al. [6]. It is optimal on a single processor. EDF's computable schedulability test was demonstrated by Baker et al. on homogeneous multiprocessors [7]. On symmetric multiprocessors, an algorithm that improves the schedulability of non-preemptive tasks was proposed by Lee et al. [8]. For the global EDF, Zhou et al. obtained a more accurate response time analysis method [9]. Jiang et al. improved the schedulability of the DAG task system [10]. Their new approach combines federated scheduling and global EDF.

Now, the field of unmanned vehicle research is very popular, and many large companies have also invested a lot of energy in the research of unmanned vehicles. The Linux kernel is used in many companies' unmanned vehicle operating systems, such as Tesla's operating system. The scheduling strategy of the Linux kernel has used scheduling algorithms such as Completely Fair Scheduling (CFS) and EDF. Thus, the study of EDF is very important.

The processor allocation strategy used in previous EDF scheduling algorithms studies on performance asymmetric multiprocessors aims to assign the high-priority tasks to the fastest processors. However, few researchers take the allocation of processors into consideration. To address this issue, we believe that it is possible to assign the high-priority tasks to a suitable or slow-speed processor.

In this paper, we study the different processor allocation strategies of EDF in performance asymmetric multiprocessor platforms. The EDF scheduling algorithm assigns priority according to the deadline of the task, and assigns the processor according to the priority. The fastest speed fit earliest deadline first (FSF-EDF) scheduling algorithm assigns high-priority tasks to the fastest processors. The best speed fit earliest deadline first (BSF-EDF) scheduling algorithm seeks to assign high-priority tasks to the most suitable speed processor. The slowest speed fit earliest deadline first (SSF-EDF) scheduling algorithm seeks to assign high-priority tasks to the slowest speed processor first. Thus, we propose a schedulability test for SSF-EDF. Below are the main contributions of this paper.

- We place three EDF scheduling algorithms with different processor allocation strategies together and propose a novel schedulability test strategy for the SSF-EDF algorithm.
- Extensive experiments are described to fully explore the performance of these three EDF scheduling algorithms.

The rest of this paper is organized as follows. Section 2 illustrates the system model and related definitions. Section 3 describes the schedulability tests for the SSF-EDF. Section 4 compares EDF scheduling algorithms with three different processor allocation strategies. Section 5 draws conclusions about the proposed strategy.

2. System Models and Definitions

2.1. Sporadic Task Systems

Let τ_i denote sporadic real-time tasks. The execution time of τ_i on the slowest processor is denoted as E_i. The relative deadline of τ_i is denoted as D_i. The minimum interval time between two task requests is denoted as T_i. Let the set of n tasks be denoted as $\Lambda = (\tau_1, \tau_2, \tau_3, \ldots, \tau_n)$. If the task satisfies the conditions of $E_i \leq D_i \leq T_i$, then Λ is considered a restricted set. The jth job of task τ_i is denoted as $\tau_{i,j}$, and its arrival time is denoted as $A_{i,j}$.

Note that an understanding of the relationship between the execution time and processor speed is critical. The worst execution time E_i is the execution time required by τ_i on the processor with speed s_1, which is a fixed value. Since we conduct research on performance asymmetric multiprocessor platforms, different processors do not have the same speed. Processors with different speeds perform differently at the same time. Therefore, the actual execution time required by τ_i on different processors is not the same. In the following, we need to take into account the effect of processor speed when calculating the execution time.

The processor utilization of task τ_i is defined as $U_i = E_i/T_i$. $U_{\max}(\Lambda)$ represents the maximum utilization among all tasks:

$$U_{\max}(\Lambda) = \max_{\tau_i \in \Lambda}(U_i). \tag{1}$$

The density of task τ_i is defined as $\delta_i = E_i/D_i$. $\delta_{\max}(\Lambda)$ represents the maximum density among all tasks:

$$\delta_{\max}(\Lambda) = \max_{\tau_i \in \Lambda}(\delta_i). \tag{2}$$

$\mathrm{DBF}(\tau_i, \Delta t)$ is used to represent the upper limit of the maximum cumulative processing time required for task τ_i's job to arrive at and within a time interval of length Δt. Among them, the size of Δt can be arbitrary. This has been proven by Baruah [11]:

$$\mathrm{DBF}(\tau_i, \Delta t) \stackrel{\text{def}}{=} \max\left(0, \left(\left\lfloor \frac{\Delta t - D_i}{T_i} \right\rfloor + 1\right) E_i\right). \tag{3}$$

The load parameters based on the DBF function are defined as follows [12]:

$$\mathrm{LOAD}(\Lambda) \stackrel{\text{def}}{=} \max_{\Delta t > 0}\left(\frac{\sum_{\tau_i \in \Lambda} \mathrm{DBF}(\tau_i, \Delta t)}{\Delta t}\right). \tag{4}$$

2.2. Performance Asymmetric Multiprocessor Platform

Let $\pi = (p_1, p_2, p_3, \ldots, p_m)$ be a performance asymmetric multiprocessor platform. The processing speed of the processors increases from small to large; that is, the processing speed of p_i is equal to or slower than that of p_{i+1}. The processor speed of the processor is denoted by s_i. For ease of understanding, the slowest processor speed is set to 1, which is $s_1 = 1$. The other, faster processors p_i are multiples of p_1, such as $s_i = 3$. The sum of the speeds of the first k processors in π is denoted as S_k, i.e.,

$$S_k(\pi) \stackrel{\text{def}}{=} \sum_{i=1}^{k} s_i. \tag{5}$$

The "lambda" parameter was defined by Funk et al. in their research on a uniform multiprocessor [13,14]:

$$\lambda(\pi) \stackrel{\text{def}}{=} \max_{1 \le i < m} \frac{\sum_{j=i+1}^{m} s_j}{s_i}. \tag{6}$$

The value range of λ is $(0, m-1]$. When all processors are the same, λ takes the maximum value $(m-1)$.

The main symbols used in this article and their meanings are summarized in Table 1.

Table 1. The symbols.

Notation	Meaning
Λ	Set of tasks
π	Asymmetric multiprocessor platform
τ_i	Task $i \in \Lambda$
$\tau_{i,j}$	The j_{th} job of task τ_i
E_i	Worst-case execution time of τ_i
D_i	Relative deadline of τ_i
T_i	Period of τ_i
p_i	i_{th} processor
s_i	Speed of the i_{th} processor
S_k	Sum of the first k processor speeds
U_i	Utilization of τ_i
δ_i	Density of τ_i
$W(t_p, t_q)$	The total amount of work done within $[t_p, t_q)$
$Q(t_p, t_q)$	The total amount of time required in $[t_p, t_q)$
J_v	The time at which v processors are executing
β	The number of carry-in jobs

2.3. SSF-EDF Scheduling

SSF-EDF assigns high priority to tasks with early deadlines. We also chose a global scheduling strategy. We establish a global ready task queue and extract tasks for scheduling according to demand. Our algorithm allows the preemption of tasks and allows tasks to be migrated between processors of different speeds. In reality, this migration introduces additional overhead. We measure this overhead by considering the number of task migrations in our experiments. The time complexity of the SSF-EDF scheduling algorithm is O(mn), where n is the number of tasks and m is the number of processors. According to the time complexity, it can be seen that SSF-EDF is used in real-time systems.

The SSF-EDF scheduling algorithm operates on a performance asymmetric multiprocessor as follows: (1) when there is an active job waiting to be executed, no processor is idle; (2) fast processors are idled when the number of processors is greater than the number of active jobs; (3) we execute higher-priority jobs on slower processors.

A given set of sporadic tasks is SSF-EDF schedulable if SSF-EDF scheduling is able to meet the deadlines of all jobs for the given set of sporadic tasks on a given platform. We will derive a schedulability test for SSF-EDF in Section 3. This can guarantee the schedulability of SSF-EDF.

Example 1. *Figure 1 is an example of SSF-EDF on the performance asymmetric multiprocessors. Task τ_i is represented by a triple (E_i, D_i, T_i). Consider $\pi = (1, 1.5)$ and $\Lambda = \{\tau_1(1.5, 1.5, 1.5), \tau_2(6, 6, 6), \tau_3(3, 6, 6)\}$. Different colors indicate different tasks. First, $\tau_{1,1}$ obtains the highest priority, $\tau_{1,1}$ executes on the slowest processor p_1, and $\tau_{2,1}$ executes on p_2. At $t = 1.5$, job $\tau_{1,1}$ is done. However, $\tau_{1,2}$ makes another request, which is still executed on p_1. Up to time $t = 4$, job $\tau_{2,1}$ is done. Since $\tau_{3,1}$ does not have a higher priority than $\tau_{1,3}$, $\tau_{3,1}$ is assigned to processor p_2 for execution until the $t = 6$ job is done. It is easy to see that under the framework of SSF-EDF, all tasks will meet their deadlines and the processor will reach* 100% *utilization.*

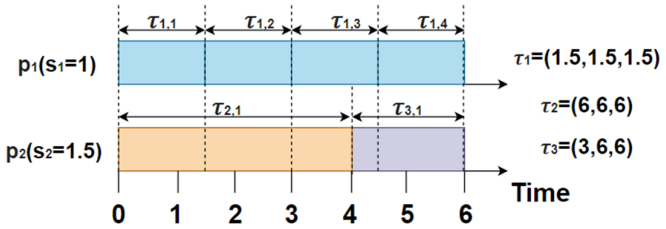


Figure 1. Task scheduling in SSF-EDF.

3. SSF-EDF Schedulability Analysis

We derive a schedulability test for SSF-EDF scheduling. Our derivation process follows the following idea: under SSF-EDF scheduling, the necessary conditions for task missed deadlines are derived. In negating this condition, the sufficient schedulability condition is obtained.

Suppose that job $\tau_{i,j}$ of task τ_i first misses the deadline at $A_{i,j} + D_i$ (see Figure 2). Under SSF-EDF scheduling, jobs with late deadlines will not affect jobs with early deadlines. Therefore, we give up the legitimate job sequence whose deadline is later than $A_{i,j} + D_i$, and only consider the SSF-EDF scheduling of the job request of the remaining legitimate sequence. Thus, τ_i missed deadline occurs at time $A_{i,j} + D_i$ (which is the earliest task missed deadline to occur).

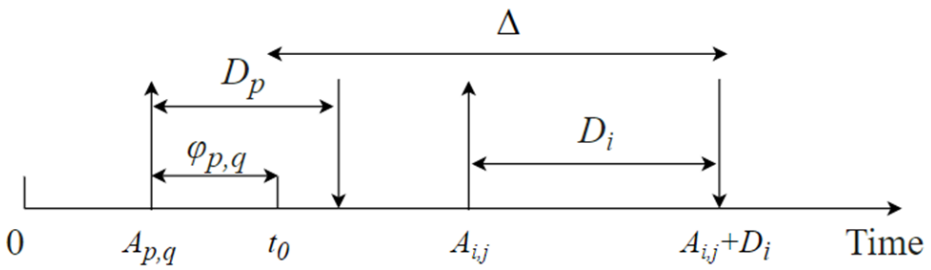


Figure 2. A job $\tau_{i,j}$ arrives at $A_{i,j}$ and misses its deadline at time $A_{i,j} + D_i$.

For a given task set Λ, let $Q(t_p, t_q)$ be the amount of processor time required within $[t_p, t_q)$, and let $W(t_p, t_q)$ be the amount of work done in $[t_p, t_q)$.

Our derivation process has the following three steps: (1) to derive the lower bound of $Q(t_0, A_{i,j} + D_i)$ in $[t_0, A_{i,j} + D_i)$, (2) to derive the upper bound of $Q(t_0, A_{i,j} + D_i)$ in $[t_0, A_{i,j} + D_i)$, (3) combining (1) and (2) yields the necessary condition. t_0 in the figure is a special time point, which we will explain later.

3.1. Lower Bound

Consider that $\tau_{i,j}$ first misses the deadline at $A_{i,j} + D_i$. Assume that under SSF-EDF scheduling, there are exactly v processors in $[A_{i,j}, A_{i,j} + D_i)$ whose execution time is denoted by J_v, $0 \le v \le m$. For example, J_1 represents the time at which only one processor is executing, and J_2 represents the time at which two processors are executing. Please note that the value of J_0 is not necessarily zero, because there are situations where all processors are idle. However, this work has already assumed that $\tau_{i,j}$ missed the deadline. Thus, within $[A_{i,j}, A_{i,j} + D_i)$, at least one processor will be processing $\tau_{i,j}$. In this case, the value of J_0 must be zero.

Above, we have mentioned the need to consider the effects of processors of different speeds. Let the execution time J_v of the v processors being executed be multiplied by the sum S_v of the speeds of these processors. We have obtained the actual workload performed by v processors through $J_v S_v$. For example, there are two tasks assigned to the processors $s_1 = 1$ and $s_2 = 2$, respectively. Let the execution time $J_2 = t$, and then the actual workload is $3t$. Then, we can write

$$W(A_{i,j}, A_{i,j} + D_i) = \sum_{v=1}^{m} S_v J_v. \tag{7}$$

Because of $\sum_{v=1}^{m} S_v J_v = S_m J_m + \sum_{v=1}^{m-1} S_v J_v$ and $S_m J_m = S_m\left(D_i - \sum_{v=1}^{m-1} J_v\right)$, we obtain

$$\begin{aligned} W\left(A_{i,j}, A_{i,j} + D_i\right) &= S_m D_i - \sum_{v=1}^{m-1} \left(S_m J_v - S_v J_v\right) \\ &= S_m D_i - \sum_{v=1}^{m-1} \frac{S_m - S_v}{s_1} s_1 J_v. \end{aligned} \tag{8}$$

As defined by $\lambda(\pi)$ in Section 2.2, when $1 \leq v \leq m-1$, it is obvious that

$$\lambda(\pi) \geq \frac{S_m - S_v}{s_1}. \tag{9}$$

From (8) and (9) above, we can draw a conclusion

$$W\left(A_{i,j}, A_{i,j} + D_i\right) \geq S_m D_i - \lambda(\pi) \sum_{v=1}^{m-1} s_1 J_v. \tag{10}$$

Due to the nature of SSF-EDF, except when all processors are busy, during time interval $\left[A_{i,j}, A_{i,j} + D_i\right)$, $\tau_{i,j}$ will be assigned to a processor with speed s_i, $s_i \geq s_1$. Then, the worst execution time $E_i \geq \sum_{v=1}^{m-1} s_1 J_v$ of $\tau_{i,j}$. Moreover, since $\tau_{i,j}$ missed the deadline, it cannot receive more execution time than E_i; otherwise, $\tau_{i,j}$ can finish before the deadline, which contradicts the previous assumption. Therefore, we obtain

$$E_i > \sum_{v=1}^{m-1} s_1 J_v. \tag{11}$$

Combining (10) and (11) gives

$$\begin{aligned} W\left(A_{i,j}, A_{i,j} + D_i\right) &> S_m D_i - \lambda(\pi) E_i, \\ \frac{W\left(A_{i,j}, A_{i,j} + D_i\right)}{D_i} &> S_m - \lambda(\pi)\delta_i, \\ \frac{W\left(A_{i,j}, A_{i,j} + D_i\right)}{D_i} &> S_m - \lambda(\pi)\delta_{\max}(\Lambda). \end{aligned} \tag{12}$$

Let

$$\mu = S_m - \lambda(\pi)\delta_{\max}(\Lambda). \tag{13}$$

From (12) and (13), it follows that

$$W\left(A_{i,j}, A_{i,j} + D_i\right) > \mu D_i. \tag{14}$$

We can find many time points t that satisfy condition $t_0 \leq A_{i,j}$. t_0 is the earliest one among all time points t that satisfy the condition. At t_0, $W\left(A_{i,j}, A_{i,j} + D_i\right) > \mu\left(A_{i,j} + D_i - A_{i,j}\right)$ still holds. This leads to $W\left(t_0, A_{i,j} + D_i\right) > \mu\left(A_{i,j} + D_i - t_0\right)$. Hence, the lower bound of $Q\left(t_0, A_{i,j} + D_i\right)$ has been deduced. Let $\Delta = A_{i,j} + D_i - t_0$. Then, we obtain

$$W\left(t_0, A_{i,j} + D_i\right) > \mu\Delta. \tag{15}$$

According to the previous assumption, job $\tau_{i,j}$ misses the deadline $A_{i,j} + D_i$. That is, the total workload W is less than or equal to the time requirement Q. It can be concluded

$$Q\left(t_0, A_{i,j} + D_i\right) \geq W\left(t_0, A_{i,j} + D_i\right) > \mu\Delta. \tag{16}$$

Now, we find that the lower bound of $Q\left(t_0, A_{i,j} + D_i\right)$ is $\mu\Delta$.

3.2. Upper Bound

Before derivation, we need to classify the job. Carry-in jobs are jobs that arrive before t_0 and are not completed before t_0. The total processor time requirement caused by these jobs is denoted by R_c. Other jobs are regular jobs. The total processor time requirement caused by this job is denoted by R_c. The sum of these two time demands is the $Q(t_0, A_{i,j} + D_i)$ that we need.

The study of Baruah et al. shows that $\Delta \times \mathrm{LOAD}(\Lambda)$ can calculate the upper bound of R_r [11,12]. LOAD and DBF are defined as follows:

$$\begin{aligned} \mathrm{DBF}(\tau_i, \Delta t) &\stackrel{\mathrm{def}}{=} \max\left(0, \left(\left\lfloor \frac{\Delta t - D_i}{T_i} \right\rfloor + 1\right) E_i\right), \\ \mathrm{LOAD}(\Lambda) &\stackrel{\mathrm{def}}{=} \max_{\Delta t > 0}\left(\frac{\sum_{\tau_i \in \Lambda} \mathrm{DBF}(\tau_i, \Delta t)}{\Delta t}\right). \end{aligned} \tag{17}$$

Let us calculate the upper bound of R_c again.

Lemma 1. *The remaining execution requirement at time t_0 for any carry-in job is lower than $\Delta\delta_{\max}(\Lambda)$.*

Proof. Let us analyze the carry-in job $\tau_{p,q}$ whose arrival time is $A_{p,q} < t_0$ and whose execution is not completed at time t_0 (see Figure 2). Let $\varphi_{p,q} = t_0 - A_{p,q}$. Let $E'_{p,q}$ be the execution time of $\tau_{p,q}$ within time interval $[A_{p,q}, t_0)$. J_v continues to use the same definition as in Section 3.1. $W(A_{p,q}, t_0)$ can be calculated from $S_m\varphi_{p,q} - \sum_{v=1}^{m-1}(S_m J_v - S_v J_v)$. Since job $\tau_{p,q}$ did not complete execution at time t_0, we can conclude

$$E'_{p,q} > S_m\varphi_{p,q} - \sum_{v=1}^{m-1}(S_m J_v - S_v J_v). \tag{18}$$

Since $W(A_{p,q}, t_0) = W(A_{p,q}, A_{i,j} + D_i) - W(t_0, A_{i,j} + D_i)$, there is

$$W(A_{p,q}, A_{i,j} + D_i) - W(t_0, A_{i,j} + D_i) = E'_{p,q}. \tag{19}$$

It is given by (18) and (19)

$$W(A_{p,q}, A_{i,j} + D_i) - W(t_0, A_{i,j} + D_i) > S_m\varphi_{p,q} - \sum_{v=1}^{m-1}(S_m J_v - S_v J_v). \tag{20}$$

According to the definition of t_0 (see (15)), it must satisfy

$$W(t_0, A_{i,j} + D_i) > \mu\Delta. \tag{21}$$

Since t_0 is the earliest time t satisfying such conditions, any time t before t_0 does not satisfy the above formula, i.e.,

$$W(A_{p,q}, A_{i,j} + D_i) \leq \mu(\Delta + \varphi_{p,q}). \tag{22}$$

We immediately have

$$\mu(\Delta + \varphi_{p,q}) - \mu\Delta > W(A_{p,q}, A_{i,j} + D_i) - W(t_0, A_{i,j} + D_i). \tag{23}$$

This is given by (20) and (23)

$$\begin{aligned} \mu(\Delta + \varphi_{p,q}) - \mu\Delta &> S_m\varphi_{p,q} - \sum_{v=1}^{m-1}(S_m J_v - S_v J_v), \\ \mu\varphi_{p,q} &> S_m\varphi_{p,q} - \sum_{v=1}^{m-1}\frac{S_m - S_v}{s_1}s_1 J_v. \end{aligned} \tag{24}$$

By combining (6), we have

$$\mu\varphi_{p,q} > S_m\varphi_{p,q} - \lambda(\pi)\sum_{v=1}^{m-1} s_1 J_v. \tag{25}$$

With the exception of the case wherein all processors are busy, the carry-in job $\tau_{p,q}$ must be executed on one of the processors. Since $\tau_{p,q}$ is not completed before t_0, the worst execution time of $\tau_{p,q}$ is not less than the execution time in $\left[A_{p,q}, t_0\right)$. Among them, the execution time of $\tau_{p,q}$ in $\left[A_{p,q}, t_0\right)$ is at least $\sum_{v=1}^{m-1} s_1 J_v$. Thus, we have

$$E'_{p,q} \geq \sum_{v=1}^{m-1} s_1 J_v. \tag{26}$$

It is given by (25) and (26) that

$$\mu\varphi_{p,q} > S_m\varphi_{p,q} - \lambda(\pi)E'_{p,q}. \tag{27}$$

It is also given by (13) and (27) that

$$\begin{aligned} (S_m - \lambda(\pi)\delta_{\max}(\Lambda))\varphi_{p,q} &> S_m\varphi_{p,q} - \lambda(\pi)E'_{p,q}, \\ E'_{p,q} &> \delta_{\max}(\Lambda)\varphi_{p,q}. \end{aligned} \tag{28}$$

Since $E_{p,q} = D_p\delta_p$, there is

$$E_{p,q} - E'_{p,q} < D_p\delta_p - \delta_{\max}(\Lambda)\varphi_{p,q}. \tag{29}$$

Because $\tau_{p,q}$ missed the deadline at $A_{i,j} + D_i$, $\tau_{p,q}$'s absolute deadline is no greater than $A_{i,j} + D_i$. We can write $D_p - \varphi_{p,q} \leq \Delta$. Thus, we have

$$\begin{aligned} E_{p,q} - E'_{p,q} &< D_p\delta_p - \delta_{\max}(\Lambda)\varphi_{p,q}, \\ E_{p,q} - E'_{p,q} &< D_p\delta_{\max}(\Lambda) - \delta_{\max}(\Lambda)\varphi_{p,q}, \\ E_{p,q} - E'_{p,q} &< (D_p - \varphi_{p,q})\delta_{\max}(\Lambda), \\ E_{p,q} - E'_{p,q} &< \Delta\delta_{\max}(\Lambda). \end{aligned} \tag{30}$$

□

Therefore, any carry-in job $\tau_{p,q}$ contributes no more to R_c than $\Delta\delta_{\max}(\Lambda)$.

Lemma 2. *The number of carry-in jobs depends on*

$$\beta = \max\{\omega : S_\omega < \mu\}. \tag{31}$$

Proof. Suppose that an arbitrarily small positive number is represented by ϵ, as shown by (21) and (22),

$$\begin{aligned} W\left(t_0, A_{i,j} + D_i\right) &> \mu\Delta, \\ W\left(t_0 - \epsilon, A_{i,j} + D_i\right) &\leq \mu(\Delta + \epsilon), \\ W\left(t_0 - \epsilon, A_{i,j} + D_i\right) - W\left(t_0, A_{i,j} + D_i\right) &< \mu\epsilon. \end{aligned} \tag{32}$$

Infer $\mu \leq S_m$ from the definition of μ. Thus, we obtain the conclusion of $W(t_0 - \epsilon, t_0) < \epsilon S_m$. Therefore, some processors are idle during $[t_0 - \epsilon, t_0)$. All the carry-in jobs are being executed. Thus, the number of carry-in jobs does not exceed β. □

Consequently, the upper bound of R_c is $\beta\Delta\delta_{\max}(\Lambda)$.

Therefore, the upper bound of $Q(t_0, A_{i,j} + D_i)$ can be expressed as

$$Q(t_0, A_{i,j} + D_i) \leq \text{LOAD}(\Lambda)\Delta + \beta\Delta\delta_{\max}(\Lambda). \tag{33}$$

3.3. SSF-EDF Schedulability Test

Theorem 1. *Sporadic task set Λ is SSF-EDF schedulable on a performance asymmetric multiprocessor platform π, provided that*

$$\text{LOAD}(\Lambda) \leq \mu - \beta\delta_{\max}(\Lambda), \tag{34}$$

where μ and β are defined in (13) and (31), respectively.

Proof. A sufficient condition for SSF-EDF schedulability arises by negating the necessary condition for $\tau_{i,j}$ missing the deadline.

From the previous discussion, the bounds for $Q(t_0, A_{i,j} + D_i)$ are as follows:

$$\begin{aligned} &Q(t_0, A_{i,j} + D_i) > \mu\Delta, \\ &Q(t_0, A_{i,j} + D_i) \leq \text{LOAD}(\Lambda)\Delta + \beta\Delta\delta_{\max}(\Lambda). \end{aligned} \tag{35}$$

From this, the necessary conditions for $\tau_{i,j}$ to miss the deadline are as follows:

$$\begin{aligned} \text{LOAD}(\Lambda)\Delta + \beta\delta_{\max}(\Lambda)\Delta &> \mu\Delta, \\ \text{LOAD}(\Lambda) &> \mu - \beta\delta_{\max}(\Lambda). \end{aligned} \tag{36}$$

Therefore, negation of (36) gives

$$\text{LOAD}(\Lambda) \leq \mu - \beta\delta_{\max}(\Lambda). \tag{37}$$

This is Theorem 1. □

4. Experimental Evaluation

In this section, we implement several schedulability tests and compare their performance through extensive simulations. In the first set of experiments, we demonstrate that BSF-EDF has the best schedulability. Then, we further verify that BSF-EDF schedules more task sets than the other two processor assignment strategies. Then, we find that as the proportion of fast processors increases, so does the proportion of the task set that the algorithm can schedule. We also compare the number of task migrations and effective processor utilization (EPU), where SSF-EDF performs best.

We randomly generate various parameters on a performance asymmetric multiprocessor platform π. The total number m of processors is randomly selected from a range of $[1, 10]$. While satisfying $q \leq m$, the number q of the slowest processors is also selected from the same range. The speed of the processor is randomly generated within $[1, 10]$. Table 2 is the parameter range for each task. We generated 10,000 task sets, and each task set contained $2(m + 1)$ tasks.

Table 2. Task parameters.

Parameter	Parameter Range
T_i	$10 \leq T_i \leq 100$
D_i	$D_i = T_i$
E_i	$0 \leq E_i \leq 50$ and $E_i \leq D_i$

Figure 3 shows that BSF-EDF performs better than FSF-EDF and SSF-EDF on four-core performance asymmetric multiprocessors. The results show that the schedulability of BSF-EDF is the highest for different system utilization cases. The X-axis represents the system utilization rate, and the system utilization rate is defined as U_{sum}/S_m. When the system utilization rate reaches 40–50%, the task set that can be scheduled by the three processor allocation strategies begins to decrease. When the system utilization reaches 80–90%, BSF-EDF can still schedule up to 93% task sets, while the other two processor allocation strategies can only schedule 81.4% and 75.1% task sets. After this, the proportion that the algorithm can schedule decreases rapidly. When the system utilization reaches 90–100%, the three processor allocation strategies can only schedule 8.9%, 0%, and 0% task sets.

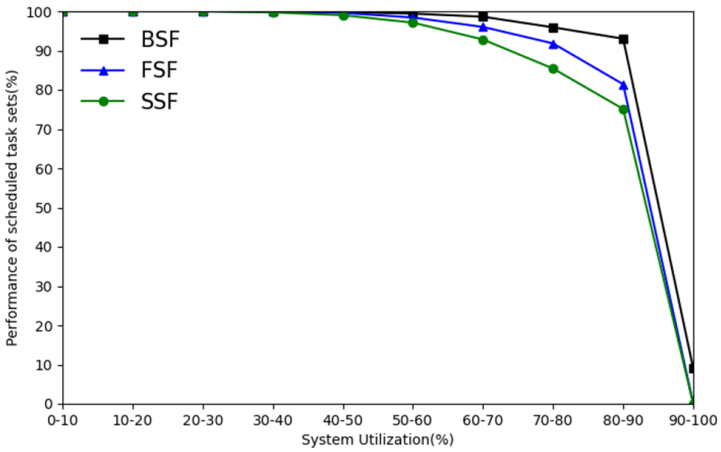


Figure 3. Performance comparison of BSF, FSF, and SSF under different system utilization values ($m = 4$).

Figure 4 shows the experimental results on seven-core performance asymmetric multiprocessors. When the system utilization is low, the three algorithms perform well. BSF-EDF performs better than the other two processor allocation strategies at high system utilization. When the system utilization reaches 80–90%, the BSF-EDF can schedule up to 88.9% of the task set, which is lower than the performance on four-core processors. However, it still schedules 10.5% and 20.8% more task sets than the other two processor allocation strategies. When the system utilization reaches 90–100%, the three processor allocation strategies can only schedule 8.7%, 0%, and 0% task sets.

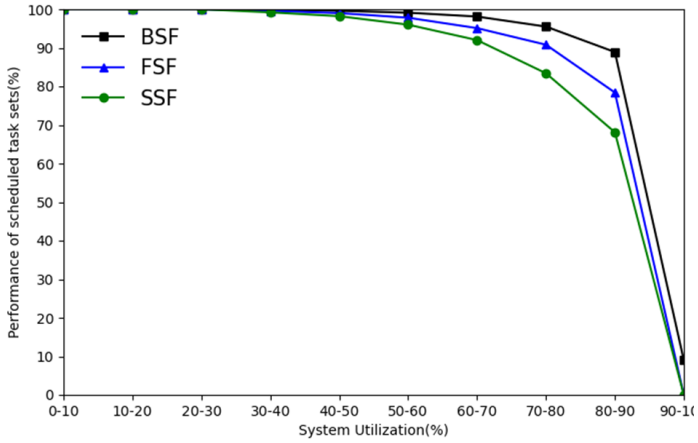


Figure 4. Performance comparison of BSF, FSF, and SSF under different system utilization values ($m = 7$).

When $m = 10$, the experiment further verifies the leading position of BSF-EDF. In Figure 5, BSF-EDF outperforms the other two processor allocation strategies at high system utilization. When the system utilization reaches 80–90%, the BSF-EDF can schedule 82.6% of the task set, which is lower than the performance on seven-core processors. However, it still schedules 9.8% and 29.5% more task sets than the other two processor allocation strategies. When the system utilization reaches 90–100%, the three processor allocation strategies can only schedule 8.3%, 0%, and 0% task sets.

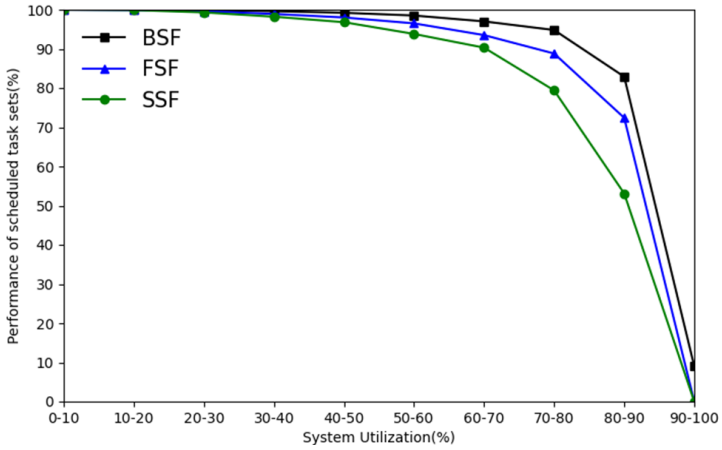


Figure 5. Performance comparison of BSF, FSF, and SSF under different system utilization values ($m = 10$).

In the first set of simulation experiments, we did not limit the number of high-utilization tasks in the task set, and the data were generated randomly. Now, we limit the proportion of high-utilization tasks in the task set, in which we define high-utilization tasks as a utilization rate of 80–100%. When the proportion of high-utilization tasks is high, the range of system utilization is bound to be limited by its definition. It is difficult to compare the performance of the three different processor allocation strategies under different system utilization values. Therefore, we choose 30% of high-utilization tasks to conduct experiments on 10-core performance asymmetric multiprocessors. In Figure 6, the performance of the three different processor allocation strategies is again degraded, but BSF-EDF still schedules more task sets than the other two processor allocation strategies. When the system utilization reaches 80–90%, the three processor allocation strategies can only schedule 73.8%, 34.2%, and 20.9% task sets. When the system utilization reaches 90–100%, the three processor allocation strategies can only schedule 6.1%, 0.9%, and 0% task sets.

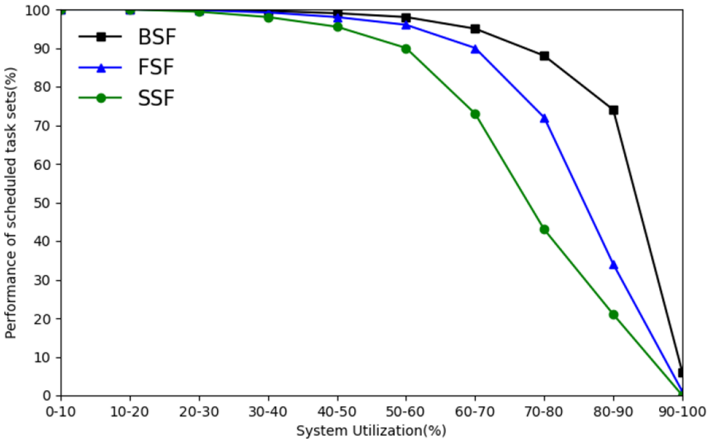


Figure 6. Performance comparison of BSF, FSF, and SSF under different system utilization values ($m = 10$).

We compare the three processor allocation strategies by modifying the proportion of fast processors on a 10-core performance asymmetric multiprocessor. In Figure 7, with the increasing proportion of fast processors, the schedulable task set ratio of the three processor allocation strategies also increases, among which BSF-EDF increases the fastest. When the fast processor ratio reaches 100%, all processors are at the same speed. In this case, the scheduling process of the three different processor allocation strategies is not different, so the three algorithms can schedule 99.9% of the task set.

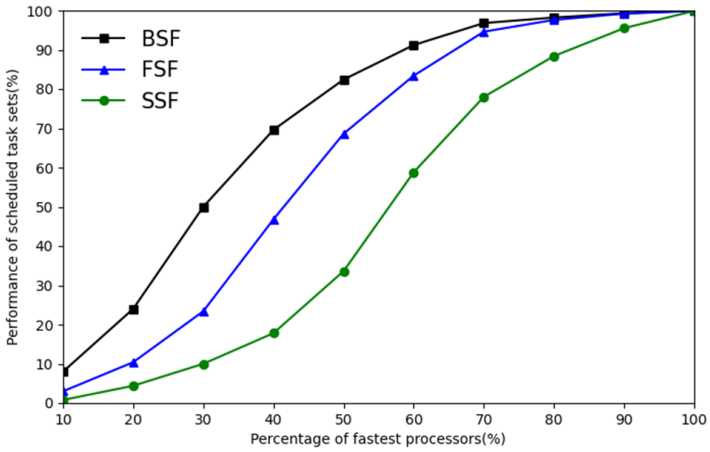


Figure 7. Performance comparison of BSF, FSF, and SSF for different percentages of fast processors ($m = 10$).

Since we allow the preemption of tasks, it is important to consider the overhead of task migration. We measure this overhead by the number of task migrations. We conduct experiments on a 10-core performance asymmetric multiprocessor. In Figure 8, the number of migrations of SSF-EDF is the least for different system utilization cases. BSF-EDF has the most migrations when the system utilization is low. When the system utilization exceeds 70–80%, FSF-EDF has the most migrations.

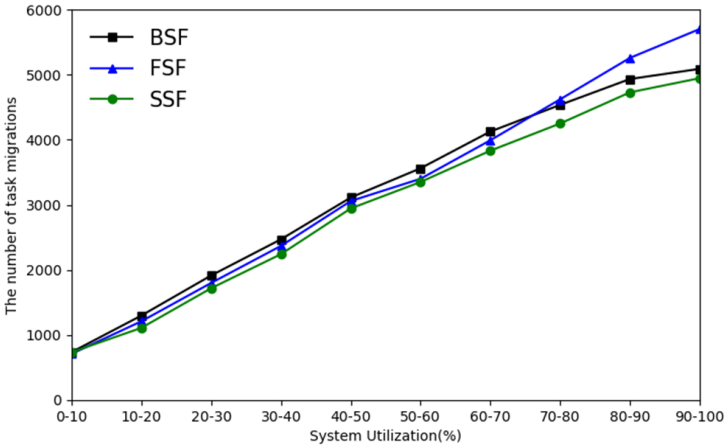


Figure 8. Comparison of task migration times of BSF, FSF, and SSF under different system utilization values ($m = 10$).

Effective processor utilization (EPU) is defined as $EPU = \sum_{i \in \Lambda} V_i / P$ [15]. If the job is completed within its deadline, the value of V is equal to the execution time of the job; if the job misses the deadline, $V = 0$. Λ is the set of tasks that the processor executes. P is the total time of execution. The level of EPU reflects the level of effective utilization of the processor by the scheduling algorithm. In Figure 9, for different system utilization rates, the EPU of SSF-EDF is the highest. This means that SSF-EDF is more efficient for processor utilization.

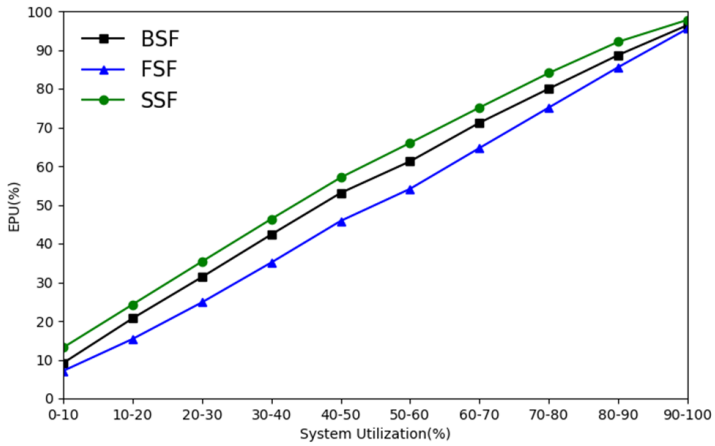


Figure 9. Comparison of EPU values of BSF, FSF and SSF under different system utilization values ($m = 10$).

5. Conclusions

We first obtain the schedulability test of SSF-EDF. Then, we study three different processor allocation strategies of the EDF scheduling algorithm on the performance asymmetric multiprocessor platform. The experiments show that BSF-EDF can schedule more task sets than the other two processor allocation strategies. SSF-EDF has the lowest number of task migrations, indicating that it has less preemption latency. SSF-EDF has the highest effective processor utilization, indicating that it utilizes the processor more efficiently.

The design of the best scheduling strategy for performance asymmetric multiprocessors still remains unsolved and is an important research problem. In this paper, inspired by the global EDF scheduling strategy, we select the best or slowest processor for EDF scheduling [16]. The purpose of this consideration is to reserve fast processors for later tasks. This provides an opportunity to improve schedulability. The simulation results illustrate that the idea of reserving fast processors with a certain strategy for subsequent tasks is useful, which is what BSF-EDF achieves. Specifically, it rationally allocates the current task according to its need for resources. In this way, the processor resources can be fully utilized while the on-time completion of the current task is guaranteed. Thus, BSF-EDF can schedule the most task sets. The number of schedulable task sets of SSF-EDF is less than that of BSF-EDF. However, SSF-EDF has the smallest number of task migrations and the highest effective processor utilization.

Author Contributions: Conceptualization, P.W. and Z.L.; methodology, P.W., Z.L. and T.Y.; software, P.W. and Z.L.; validation, P.W., Z.L. and T.Y.; formal analysis, P.W. and Z.L.; investigation, P.W., Z.L., T.Y. and Y.L.; resources, P.W., T.Y. and Y.L.; data curation, P.W. and Z.L.; writing—original draft preparation, P.W. and Z.L.; writing—review and editing, P.W., Z.L., T.Y. and Y.L.; visualization, P.W., Z.L. and Y.L.; supervision, P.W. and Z.L.; project administration, P.W., Z.L., T.Y. and Y.L; funding acquisition, P.W., T.Y. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (No. 62006146, 62001143, and 62002210) and the National Natural Science Foundation of Shandong Province, China (No. ZR2020QF006).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bertout, A.; Goossens, J.; Grolleau, E.; Poczekajlo, X. Template schedule construction for global real-time scheduling on unrelated multiprocessor platforms. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 216–221.
2. Derafshi, D.; Norollah, A.; Khosroanjam, M.; Beitollahi, H. HRHS: A High-Performance Real-Time Hardware Scheduler. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 897–908. [CrossRef]
3. Selim, Z.; El-Attar, N.E.; Ghoneim, M.E.; Awad, W.A. Performance Analysis of Real-Time Scheduling Algorithms. In Proceedings of the ICICSE '20: 2020 International Conference on Internet Computing for Science and Engineering, Male, Maldives, 14–16 January 2020; pp. 70–75.
4. Irtija, N.; Anagnostopoulos, I.; Zervakis, G.; Tsiropoulou, E.E.; Amrouch, H.; Henkel, J. Energy Efficient Edge Computing Enabled by Satisfaction Games and Approximate Computing. *IEEE Trans. Green Commun. Netw.* **2022**, *6*, 281–294. [CrossRef]
5. Mahmood, B.; Ahmad, N.; Khan, M.I.; Akhunzada, A. Dynamic Priority Real-Time Scheduling on Power Asymmetric Multicore Processors. *Symmetry* **2021**, *13*, 1488. [CrossRef]
6. Liu, C.L.; Layland, J.W. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM* **1973**, *20*, 40–61. [CrossRef]
7. Baker, T.P. Multiprocessor EDF and deadline monotonic schedulability analysis. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Washington, DC, USA, 3–5 December 2003; pp. 120–129.
8. Lee, H.; Lee, J. Limited Non-Preemptive EDF Scheduling for a Real-Time System with Symmetry Multiprocessors. *Symmetry* **2020**, *12*, 172. [CrossRef]
9. Zhou, Q.; Li, G.; Zhou, C.; Li, J. Limited Busy Periods in Response Time Analysis for Tasks Under Global EDF Scheduling. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020,** *40*, 232–245. [CrossRef]
10. Jiang, X.; Sun, J.; Tang, Y.; Guan, N. Utilization-Tensity Bound for Real-Time DAG Tasks under Global EDF Scheduling. *IEEE Trans. Comput.* **2020**, *69*, 39–50. [CrossRef]
11. Baruah, S.K.; Mok, A.K.; Rosier, L.E. Preemptively scheduling hard-real-time sporadic tasks on one processor. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Lake Buena Vista, FL, USA, 5–7 December 1990; pp. 182–190.
12. Fisher, N.; Baker, T.P.; Baruah, S.K. Algorithms for Determining the Demand-Based Load of a Sporadic Task System. In Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06), Sydney, Australia, 16–18 August 2006; pp. 135–146.
13. Funk, S.; Goossens, J.; Baruah, S.K. On-line Scheduling on Power asymmetric Multiprocessors. In Proceedings of the Real-Time Systems Symposium (RTSS), London, UK, 2–6 December 2001; pp. 183–192.
14. Funk, S.H. EDF Scheduling on Heterogeneous Multiprocessors. PhD Thesis, Department of Computer Science, The University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2004.
15. Shah, A.; Kotecha, K.; Shah, D. Dynamic scheduling for real-time distributed systems using ACO. *Int. J. Intell. Comput. Cybern.* **2010**, *3*, 279–292. [CrossRef]
16. Baruah, S.K.; Goossens, J. The EDF Scheduling of Sporadic Task Systems on Power asymmetric Multiprocessors. In Proceedings of the Real-Time Systems Symposium, Barcelona, Spain, 30 November–3 December 2008; pp. 367–374.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.