*sustainability*

MDPI

*Article*

# Energy-Saving Task Scheduling Based on Hard Reliability Requirements: A Novel Approach with Low Energy Consumption and High Reliability

**Qingfeng Chen, Yu Han, Jing Wu * and Yu Gan**

School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China; chqhong@wust.edu.cn (Q.C.); hanyu_1025@163.com (Y.H.); rorchach2k@gmail.com (Y.G.)
* Correspondence: wujingecs@wust.edu.cn

**Abstract:** With the increasing complexity of application situations in multi-core processing systems, how to assure task execution reliability has become a focus of scheduling algorithm research in recent years. Most fault-tolerant algorithms achieve hard reliability requirements through task redundancy, which increases energy consumption and contradicts the concept of sustainable development. In this paper, we propose a new algorithm called HDFE (Heterogeneous-Dag-task-fault-tolerance-energy-efficiency algorithm) that combines DVFS technology and task replication technology to solve the scheduling problem of DAG applications concerning energy-saving and hard reliability requirements in heterogeneous multi-core processor systems. Our algorithm is divided into three phases: the priority calculation phase, the task replication phase, and the task assignment phase. The HDFE algorithm achieved energy savings while meeting hard reliability requirements for applications, which was based on the interrelationship between reliability and energy consumption in filtering task replicas. In the experimental part of this paper, we designed four comparison experiments between the EFSRG algorithm, the HRRM algorithm, and the HDFE algorithm. The experimental results showed that the energy consumption of task scheduling using the HDFE algorithm is lower than other algorithms under different scales, thus achieving energy savings and complying with the concept of sustainable development.

**Keywords:** heterogeneous systems; scheduling algorithm; fault tolerance



**Citation:** Chen, Q.; Han, Y.; Wu, J.; Gan, Y. Energy-Saving Task Scheduling Based on Hard Reliability Requirements: A Novel Approach with Low Energy Consumption and High Reliability. *Sustainability* **2022**, *14*, 6591. https://doi.org/10.3390/su14116591

Academic Editor: James (Jong Hyuk) Park

Received: 12 April 2022
Accepted: 24 May 2022
Published: 27 May 2022



## 1. Introduction

### 1.1. Background

Since the late 1990s, with the development of semiconductor technology, the number of transistors on a single chip has greatly increased, while multi-core processors have also seen a lot of development [1]. Artificial intelligence applications have also gradually penetrated various industries such as intelligent medicine, autonomous driving, aerospace, and security surveillance, owing to the powerful computing power brought by multi-core processors. These industries have rigid requirements for the reliability of task execution. On the other hand, energy savings is also an eternal research topic with regard to embedded systems. In the field of task scheduling algorithms in multi-core processors, the discussion about the improvement of energy savings and reliability has never been interrupted.

### 1.2. Motivation

Task replication techniques are widely used in fault-tolerant scheduling algorithms for multi-core systems. However, redundant task replicas cause a surge in energy consumption while guaranteeing the reliability of task execution [2]. Therefore, how to save energy in scheduling algorithms based on task replication techniques has become a concern for scholars. Considering energy savings, the DVFS technique is undoubtedly the best choice. The DVFS technique achieves energy savings by reducing the processing frequency of

processor cores. Existing studies [3,4] have shown that the failure rate increases when the processor cores are at a low frequency, which easily leads to task execution failure and thus reduces the system's reliability. There are some algorithms that have been proposed, which combine task replication techniques with DVFS techniques to satisfy both application reliability requirements and energy efficiency. For example, the EFSRG algorithm, by filtering application replicas in an ascending order of energy consumption, allows the reliability of the application to be satisfied [5]. In the process of screening task replicas, such scheduling algorithms take the energy consumption of task replicas as the ranking basis and the reliability of task replicas as the condition constraint; this is accomplished without considering the comprehensive performance of task replicas in the screening process. Therefore, for this paper, we considered how to take into account the comprehensive performance of task replicas in the screening process of task replicas, as it is possible to find the task replica that meets the task reliability requirements and has lower energy consumption.

### *1.3. Contribution*

In this paper, we focus on the problem of scheduling computational tasks with reliability and security requirements in heterogeneous multi-core processors for various domains, while considering energy savings. We consider task replication techniques, which guarantee the successful execution of a task by duplicating application replicas; we also ensure the continuation of an application as the execution of a task fails due to a transient failure of the processor. Each core in a multi-core processor system can be independently frequency-regulated by the DVFS technique, and we consider regulating the execution frequency of task replicas in order to achieve energy savings. The main contributions of this paper are the following:

1. In this paper, we propose a static heuristic scheduling algorithm, HDFE, for solving the scheduling problem of DAG applications with energy-saving requirements and hard reliability requirements in heterogeneous multi-core processor systems. The algorithm is divided into three phases: the priority calculation phase, the task replication phase, and the task assignment phase.
2. In the task priority calculation stage, this paper presents a new priority calculation method based on the execution time uncertainty of tasks.
3. This paper proposes a task replica screening method that combines the comprehensive performance of reliability and energy consumption in the task replication phase.
4. In the experimental part, the simulation experiments designed for this paper compare the differences in energy consumption between the EFSRG algorithm [5], the HRRM algorithm [6], and the HDFE algorithm proposed in this paper and are based on simulation scheduling experiments on an actual application and a random application.

The rest of the paper is organized as follows. Section 2 introduces related work, Section 3 introduces related models, Section 4 presents the algorithms and the cases, and Section 5 describes the experiments. Finally, in Section 6, we discuss our conclusions.

## 2. Related Work

This section discusses the key research works related to energy-saving task scheduling and reliability task scheduling.

Energy consumption is an extremely important evaluation property of scheduling algorithms; the discussion on energy-saving scheduling algorithms has never stopped in various environments such as single-core processor systems, multi-core processor systems, real-time systems, or non-real-time systems. Most of the discussions on energy savings in heterogeneous processor systems rely on techniques such as VFI, DPM, DVS, and DVFS to dynamically regulate the voltage or frequency of processor cores in order to achieve energy savings [7]. The scheduling algorithm proposed in [8] aimed to achieve energy savings by dynamically adjusting processor voltage using the DVS technique. The DPM algorithm mainly achieves static energy consumption reduction by adjusting the processor

state [9]. The VFI technique, on the other hand, discusses energy savings by regulating the voltage of processor cores in groups within processor systems that have a large number of cores [10–12]. The DVFS technique is most widely used in energy-efficient scheduling algorithms [13–15]. However, the DVFS technique also has a drawback in that the transient failure rate increases when the processor core is running at low frequencies, which leads to a higher probability of task execution failure. This drawback of the DVFS technique causes a conflict between its energy-saving capability and its reliability. How to explore a highly reliable and low energy consumption scheduling algorithm in the conflict between has thus become the focus of research.

Most of the research on fault-tolerant scheduling discusses how to ensure that the final execution results of tasks are not affected in case of transient or permanent processor failures. Many factors can cause processor failures, including high temperature, hardware failures, etc. [16,17]. The study of fault-tolerant scheduling mainly considers transient failures because the probability of permanent failures during task execution is much lower than that of transient failures [18]. Existing fault-tolerant scheduling algorithms generally quantify the probability of successful task execution by defining it as the reliability of the task. In this regard, many fault-tolerant techniques have been proposed in order to achieve the task reliability requirement; these include the task replication [19,20], primary-backup [21], and checkpoint [22] techniques.

Niraj Kumar et al. in [2] discussed the reliable energy-efficient scheduling of non-preemptive real-time tasks in heterogeneous multi-core processing systems, but the diverse ideas presented in their discussion about heterogeneous multi-core systems is reflected in the fact that each core can independently adjust its operating frequency without discussing the heterogeneity of the hardware parameters of the processor cores.

The HRRM algorithm proposed in [6] considers the hard reliability requirements of an application through task replication, but it does not consider energy savings while achieving the reliability requirement.

The EFSRG algorithm was proposed in [5] to satisfy the hard reliability requirements of tasks by task replication while achieving energy savings by regulating the execution frequency of task replicas. However, as stated in the previous section, the EFSRG algorithm does not consider the energy consumption and reliability of task replicas together when screening the task replicas, and it does not fully utilize the interrelationship between energy consumption and reliability.

In this paper, we will discuss the dynamic scheduling algorithm for DAG applications in heterogeneous multi-core systems based on the relationship between the energy consumption and the reliability of task execution, as well as the energy saved while satisfying hard reliability requirements.

## 3. System Model

### *3.1. Task Model*

As shown in Figure 1, an application is represented by a DAG G. Let G = ($V$, $E$), where $V$ is a set of $v$ tasks, which is represented by $V = \{v_1, v_2, \ldots, v_n\}$, and $E$ is a set of directed edges among tasks. Each edge $e_{i,j}$ represents task $v_i$, dependency constraints between $v_j$, i.e., task $v_i$ must finish its execution and transfer the resulting data to solve the data dependency before task $v_j$ starts. The weight of each edge $e_{i,j}$ represents the communication cost between task $v_i$ and $v_j$ and is denoted by $cp_{i,j}$.

As a supplement to DAG, W is a matrix of n * m, n is the number of tasks in the application, and m is the number of processors in this processor system. $w_{i,k}$ represents the estimated execution time of $task_k$ on processor $u_k$.

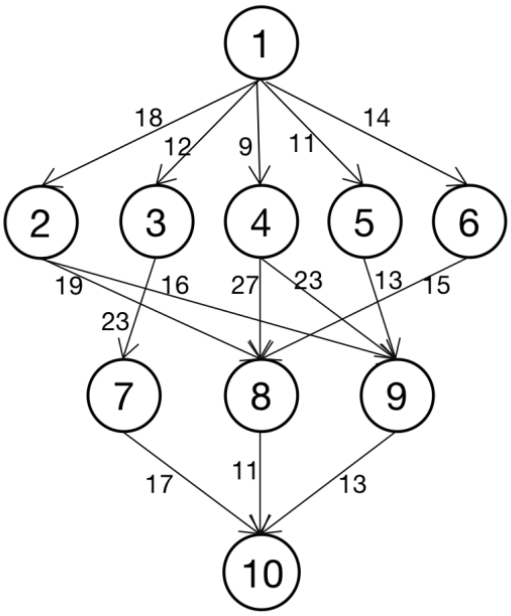



**Figure 1.** DAG diagram shows the application.

### *3.2. Processor Model*

This paper adopts a heterogeneous multi-core processor structure, and the cores in the processor are connected together by a bus, as shown in Figure 2. The processor is expressed as $U = \{u_1, u_2, \ldots, u_m\}$, $u_k(\alpha_k, \lambda_k, d_k, c_k, P_{ind,k}, P_{s,k}, f_{max,k}, f_{min,k}) \in U$ represents a core of a processor, where $c_k$ indicates the conversion capacitance of the processor, which is a hardware parameter. $\alpha_k$ indicates the dynamic power index, while $P_{ind,k}$, $P_{s,k}$ indicates frequency-dependent dynamic power and frequency-independent static power. $f_{max,k}$, $f_{min,k}$ indicates the maximum and minimum frequencies of processor core operation. According to DVFS technology, the processor core operating frequency can be adjusted between the minimum and maximum frequencies. $\lambda_k$ indicates the processor core failure rate, $d_k$ is a static parameter related to the hardware. The above parameters will be used in the reliability model and the energy consumption model in the following sections. The related parameters of the processor structure in Figure 2 are shown in Table 1.

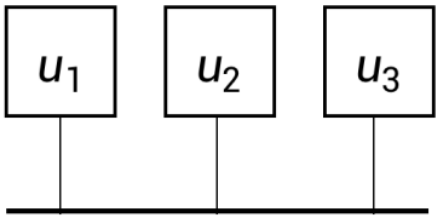



**Figure 2.** Three-core processor architecture.

**Table 1.** Hardware parameters.

| | $P_{ind}$ | $P_s$ | c | $\alpha$ | d | $\lambda$ | $f_{max}$ | $f_{min}$ |
|---|---|---|---|---|---|---|---|---|
| $u_1$ | 0.03 | 0.005 | 0.8 | 2.9 | 2.0 | 0.00020 | 1 | 0.2583 |
| $u_2$ | 0.03 | 0.005 | 0.9 | 2.9 | 2.0 | 0.00013 | 1 | 0.2480 |
| $u_3$ | 0.03 | 0.005 | 1.0 | 3.0 | 2.0 | 0.00050 | 1 | 0.2466 |

### *3.3. Reliability Model*

Due to hardware failure, high temperature, and other unknown reasons, the task may fail to complete. This paper proposes a reliability model based on transient failure probability, which is based on previous reliability scheduling research. The model uses the exponential function based on execution time and fault rate to express mission reliability.

When a task is executed, the fault rate can be expressed by $\lambda$, and fault rate is only related to the hardware parameter. When the processor is executed at the frequency $f$, the fault rate can be expressed as follows:

$$\lambda(k, f) = \lambda_k \times 10^{\frac{d_k(1-f)}{1-f_{min,k}}} \tag{1}$$

where $\lambda_k$ is the fault rate executed at frequency $f_{max,k}$ of the processor $u_k$; $d_k$ is a static parameter related to the hardware, reflecting the sensitivity of the probability of failure to frequency changes; and $f_{min,k}$ is the minimum frequency of $u_k$. Furthermore, the reliability of the task replica could be expressed by $r$, and the reliability of task $v_i$ executed at frequency $f_{i,k}$ on the processor $u_k$ is expressed as follows:

$$r(i, k, f_{i,k}) = e^{-\lambda\left(k, f_{i,k}\right) \times w_{i,k} \times \frac{f_{max,k}}{f_{i,k}}} \tag{2}$$

In particular, when $f_{i,k} = 0$, it means that the task has no replica on this processor, $r(i, k, f_{i,k}) = 0$. The reliability of task $v_i$ is calculated by the reliability of each replica of the task, which is expressed as:

$$r_{actual}(i) = 1 - \prod_{k=1}^{m}(1 - r(i, k, f_{i,k})) \tag{3}$$

Therefore, the reliability of the application could be represented as follows:

$$R_{total} = \prod_{i=1}^{n} r_{actual}(i) \tag{4}$$

Our target is to let

$$R_{total} \geq R_{req} \tag{5}$$

Since the reliability $R_{total}$ given by the application represents the reliability of the entire application, and each task is dynamically allocated both processor and operating frequencies during the scheduling process, $R_{total}$ needs to be decomposed into task reliability requirements $r_{req}(i)$, expressed as:

$$\begin{aligned} & r_{req}(i) \\ & = \frac{R_{req}}{\prod_{j \in allocated(i)} r_{actual}(j) \times \prod_{j \in unallocated(i)} r_{req}(j)} \\ & = \frac{R_{req}}{\prod_{j \in allocated(i)} r_{actual}(j) \times \prod_{j \in unallocated(i)} \sqrt[n]{R_{req}}} \end{aligned} \tag{6}$$

where allocated $(i)$ represents the applications that have been allocated before task $v_i$ in the priority queue, and unallocated $(i)$ represents the applications that have not been allocated after task $v_i$ in the priority queue. Then, if each task $v_i$ in the application satisfies $r_{actual}(i) \geq r_{req}(i)$, then the inequality (5) can be satisfied, that is, the reliability requirement of the application can be satisfied.

### *3.4. Power Model*

The power consumption of a processor is mainly composed of frequency-related dynamic consumption, frequency-independent dynamic consumption, and static con-

sumption. Among these, the frequency-related dynamic power consumption is the main component, and the total power of the processor is represented by $P$; thus, the power model is expressed as follows:

$$\begin{aligned} P(k,f) &= P_{s,k} + g(P_{ind,k} + P_{d,k}) \\ &= P_{s,k} + g\left(P_{ind,k} + c_k \times f^{\alpha(k)}\right) \end{aligned} \tag{7}$$

$P_{s,k}$ means frequency-independent static power, $P_{ind,k}$ means frequency-independent dynamic power, $P_{d,k}$ means frequency-dependent dynamic power, g means system state, $g = 0$ when the system is in sleep, $g = 1$ when the system is running, and $c_k$ means the switching capacitance of the processor, which is a hardware parameter, and $\alpha_k$ represents the dynamic power exponent. Furthermore, the dynamic energy consumption of the replicated task $v_i$ executed on the processor $u_k$ at frequency $f_{i,k}$ can be expressed as follows:

$$E_d(i,k,f_{i,k}) = (P_{ind,k} + c_k \times f^{\alpha_k}) \times w_{i,k} \times \frac{f_{max,k}}{f_{i,k}}. \tag{8}$$

The scheduling dynamic energy consumption of task $v_i$ is calculated by the dynamic energy consumption of each replica of the task.

$$E_{d\_actual}(i) = \sum_{k=1}^{m} E_d(i,k,f_{i,k}) \tag{9}$$

The static energy consumption of application is related to the $SL$ (schedule length) of the task.

$$E_s = SL \times P_s \tag{10}$$

The total energy consumption of the application is expressed as follows:

$$E_{total} = \sum_{i=1}^{n} E_{d\_actual}(i) + E_s. \tag{11}$$

In particular, when $f_{i,k} = 0$, it means that the task has no replica on this processor, $E_d(i,k,f_{i,k}) = 0$.

## 4. Algorithm

Our algorithm consists of three parts: task priority calculation, task replication, and task scheduling. In the task priority calculation stage, the task priority sequence of the application is obtained, and in the task replication stage, the reliability requirements of the tasks are first calculated according to the task priority sequence. Then, the tasks are copied according to the determined reliability requirements. Finally, a group of task replicas that meet the requirements and save energy is screened out. In the task scheduling stage, the scheduling scheme of tasks in the application and in the task replica of the processor is obtained according to the previous two stages.

### *4.1. Calculation of Task Priority*

Before replicating tasks in the application, its reliability requirements should be determined first. According to Formula (3), which was derived in the reliability model, we know that the reliability requirement of tasks is related to the processing order of tasks in the application, so we need to give the method of calculating tasks priority in the first step of the scheduling algorithm to determine the priority sequence of the tasks.

Because our scheduling algorithm will combine DVFS technology to scale the execution frequency of the tasks, the execution time of tasks on the processor cannot be determined in the priority calculation stage. Therefore, in the task priority calculation stage, we ignore the influence of task execution time and focus on the influence of communication cost between tasks.

We use the rank value to express the ranking weight of tasks, and the rank is calculated as shown in Formula (12). The priority sequence can be obtained by ranking tasks in descending order of rank value.

$$Rank(v_i) = \max_{v_j \epsilon succ_{(vi)}} \left( Rank(v_j) + cp_{i,j} \right) \tag{12}$$

Taking the application in Figure 1 as an example, the task sequences calculated by Formula (12) are $v_1$, $v_3$, $v_4$, $v_2$, $v_5$, $v_6$, $v_7$, $v_9$, $v_8$, and $v_{10}$.

### *4.2. Task Replication*

In the previous section, we obtained the priority sequence of tasks. In the task replication stage, we first need to decompose the reliability requirements of the application into the reliability requirements of the tasks, and then replicate the tasks according to their reliability requirements. The algorithm for the task replication phase is shown in Algorithm 1.

The first function of the task replication phase is to take out the first task from the priority sequence $v_i$ according to the reliability requirements in the reliability model, then perform Equation (3) to calculate the $r_{req}(i)$ reliability requirements, as shown in line 3 of Algorithm 1.

Taking task $v_1$ as an example, the calculation of the reliability requirements is as follows:

$$r_{req}(1) = \frac{0.95}{(\sqrt[10]{0.95})^9} = 0.994884$$

After determining the reliability requirements of the tasks, how to use task replication to meet the reliability requirements has become a problem to be considered. According to the research in [2], given the task reliability requirements, the number of task replicas and the execution frequency of the task replicas are negatively correlated; that is, if the number of task replicas increases, the minimum execution frequency of each task replica will decrease. Therefore, the key to task replication is how to select the number of replicas and the frequency of the task replicas.

**Algorithm 1:** Task replication phase.

**Input:** $G(N, E), W, R_{req}, U, task\_queue$
**Output:** frequency[]
1: **while** (*task_queue* is not empty) **do**
2: $v_i \leftarrow$ first task in *task_queue*
3: $r_{req}(i) \leftarrow$ calculated from Equation (3)
4: while($r_{actual}(i) < r_{req}(i)$) do
5: $f(i,k) \leftarrow f_{\text{low,k}}$
6: **for** k = 1:m **do**
7: $f(i, min)$ find the min $\varphi(i, k, f_{i,k})$
8: **end for**
9: frequency$[i, min] \leftarrow f(i, min)$;
10: update the value of $r_{actual}(i)$
11: end while
12: end while
13: return frequency[]

Because the hardware parameters of each core in the processor system being considered are different, it is impossible to calculate the execution frequency of each replica when determining the task reliability requirements and the number of replicas. We assume that the task has a task replica on each core, and that the initial execution frequency is 0. Search for the task execution frequency on each core until we find the replica combination that meets the requirements of task reliability. In the final result, if the execution frequency of a

task replica $f_{i,k}$ is equal to 0, it means that there is no replica of the task $v_i$ on the processor $u_i$.

If energy savings is not considered, we can always choose the most reliable replica for the next iteration when traversing the execution frequency of task replicas on each core. However, with regard to energy savings, we need to define a comprehensive index that includes reliability and energy consumption, and which is used to screen the task replicas that are able to enter the next iteration. For a group of task replicas that have finally been screened out, we hope it can achieve high reliability with low energy consumption. Hence, we take the energy consumption required to achieve the unit reliability goal as our comprehensive parameter, and the calculation method applied is as follows:

$$\varphi(i,k,f_{i,k}) = E_d(i,k,f_{i,k})/r(i,k,f_{i,k}) \tag{13}$$

According to Formulas (2) and (8),

$$\varphi(i,k,f_{i,k}) = \frac{(P_{ind,k} + C_k \times f_{i,k}\alpha_k) \times w_{i,k} \times \frac{f_{max,k}}{f_{i,k}}}{e^{-\lambda(k,f_{i,k}) \times w_{i,k} \times \frac{f_{max,k}}{f_{i,k}}}} \tag{14}$$

We assume that the task has a task replica on each core. In the process of task replica screening, our algorithm selects each task replica $\varphi$, a replica with the lowest value, so that its frequency is increased; the reliability of tasks is calculated according to Formula (3). If the reliability requirement is met, this group of task replicas will be output; otherwise, the screening process will continue. This is shown in lines 5–9 of Algorithm 1.

Figure 1 shows the tasks in the application $v_i$. An example of the screening process for task replicas is shown in Table 2.

**Table 2.** Tasks $v_1$—task replica filtering process.

| | $u_1$ | $u_2$ | $u_3$ |
|---|---|---|---|
| $f_{1,k}$ | 0.2583 | 0.2480 | 0.2466 |
| $\varphi(i,k,f_{i,k})$ | 3.4514 | 3.8598 | 2.9508 |
| frequency[1,$k$] | 0 | 0 | 0.2466 |
| Substitute $f_{1,3}$ in the frequency $\varphi(1,3,f_{1,3})$ as the smallest of the three, determine $r_{actual}(1) = 0.5565 < r_{req}(1)$, and continue the iteration while $f_{1,3}$ is increased by 0.01. | | | |
| $f_{1,k}$ | 0.2583 | 0.2480 | 0.2566 |
| $\varphi(i,k,f_{i,k})$ | 3.4514 | 3.8598 | 2.8165 |
| frequency[1,$k$] | 0 | 0 | 0.2566 |
| Substitute $f_{1,3}$ in the frequency $\varphi(1,3,f_{1,3})$ as the smallest of the three, determine $r_{actual}(1) = 0.5840 < r_{req}(1)$, and continue the iteration while $f_{1,3}$ is increased by 0.01. | | | |
| ..... | | | |
| $f_{1,k}$ | 0.4283 | 0.2480 | 0.5666 |
| $\varphi(i,k,f_{i,k})$ | 3.5869 | 3.8598 | 3.5686 |
| frequency[1,$k$] | 0 | 0 | 0.5666 |
| Substitute $f_{1,3}$ in the frequency $\varphi(1,3,f_{1,3})$ as the smallest of the three, determine $r_{actual}(1) = 0.9951 > r_{req}(1)$, then end the iteration and output the result. | | | |

### *4.3. Task Scheduling Stage*

In the task scheduling phase, we need to schedule the tasks based on the previously obtained task priority queue from task replication scheme. Tasks are scheduled according to the following rules.

1. The task replica has the same priority as the primary task.
2. When all replicas of the task have successfully been executed, its child nodes can start to receive the data of the task.

3. The task cannot be preempted during execution.

Taking the application in Figure 1 as an example, the final scheduling result obtained by our scheduling algorithm is shown in Table 3. AST is the start execution time of the task replica on the corresponding core, and AFT is the end execution time.

**Table 3.** Application task replication results.

| | | $u_1$ | $u_2$ | $u_3$ | $r_{actual}$ | $E_{actual}$ |
|---|---|---|---|---|---|---|
| $v_1$ | frequency | 0.428323974 | / | 0.566621207 | 0.995 | 6.583 |
| | AST | 0 | / | 0 | | |
| | AFT | 32.685 | / | 15.883 | | |
| $v_2$ | frequency | 0.608323974 | 0.288042393 | / | 0.995 | 8.272 |
| | AST | 106.498 | 78.840 | / | | |
| | AFT | 127.868 | 144.802 | / | | |
| $v_3$ | frequency | 0.468323974 | 0.348042393 | / | 0.995 | 5.482 |
| | AST | 32.685 | 27.884 | / | | |
| | AFT | 56.173 | 65.235 | / | | |
| $v_4$ | frequency | 0.258323974 | 0.588042393 | / | 0.997 | 5.338 |
| | AST | 56.173 | 65.235 | / | | |
| | AFT | 106.498 | 78.840 | / | | |
| $v_5$ | frequency | 0.448323974 | 0.368042393 | / | 0.995 | 5.705 |
| | AST | 127.868 | 144.802 | / | | |
| | AFT | 154.634 | 180.124 | / | | |
| $v_6$ | frequency | 0.438323974 | / | 0.526621207 | 0.994 | 6.068 |
| | AST | 154.634 | / | 15.884 | | |
| | AFT | 184.293 | / | 32.974 | | |
| $v_7$ | frequency | 0.748323974 | 0.248042393 | / | 0.999 | 6.278 |
| | AST | 184.293 | 180.124 | / | | |
| | AFT | 193.647 | 240.598 | / | | |
| $v_8$ | frequency | 0.658323974 | / | / | 0.993 | 2.035 |
| | AST | 263.327 | / | / | | |
| | AFT | 270.922 | / | / | | |
| $v_9$ | frequency | 0.258323974 | 0.608042393 | / | 0.995 | 7.979 |
| | AST | 193.647 | 240.598 | / | | |
| | AFT | 263.327 | 260.333 | / | | |
| $v_{10}$ | frequency | / | 0.628042393 | / | 0.992 | 2.937 |
| | AST | / | 281.922 | / | | |
| | AFT | / | 293.068 | / | | |
| $E_{total} = 56.68 + 2.98 = 59.66$, $R_{total} = 0.9501$, $SL = 293.068$ | | | | | | |

## 5. Experiment

For this paper, we used MATLAB, a mighty data processing tool, to conduct simulation experiments in task scheduling. Since the EFSRG algorithm and HRRM algorithm are similar to the models in this paper, we will compare the performance of the EFSRG algorithm, HRRM algorithm, and HDFE algorithm in the experiments. The range of the relevant parameters of the processor in the simulation experiments is shown in Table 4.

**Table 4.** Relevant parameter range of the processor in the simulation experiment.

| Parameter | Lower Limit | Upper Limit |
|---|---|---|
| $P_{ind}$ | 0.03 | 0.07 |
| $P_s$ | $1 \times 10^{-3}$ | $5 \times 10^{-3}$ |
| c | 0.8 | 1.2 |
| $\alpha$ | 2.5 | 3.0 |
| d | 1 | 3 |
| $\lambda$ | $1 \times 10^{-4}$ | $1 \times 10^{-3}$ |

### *5.1. FFT (Fast Fourier Transform) Application Scheduling Experiment*

The Fourier transform is mainly used in signal processing to convert time domain signals to frequency domain signals. The Fast Fourier Transform is an efficient algorithm for computing the discrete Fourier transform in a computer. The use of the Fast Fourier Transform allows the number of multiplications required for the discrete Fourier transform to be greatly reduced. As shown in Figure 3, the dependencies between tasks when using the Fast Fourier Transform with four size of input vectors are plotted. The number of tasks $n$ in the Fast Fourier Transform application is related to the number of size of input vectors $N$, as shown in Equation (15) [23], where $N = 2\rho$.

$$n = (2 \times N - 1) + N \times \log(N) = (2 + \rho) \times 2^{\rho} - 1, \rho \in Z^{+} \quad (15)$$

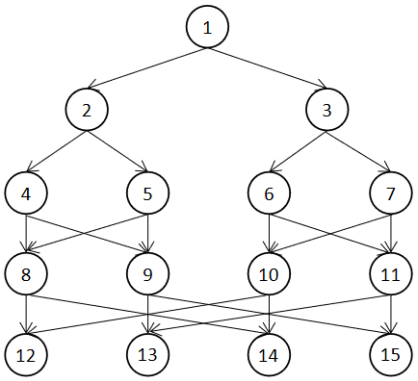
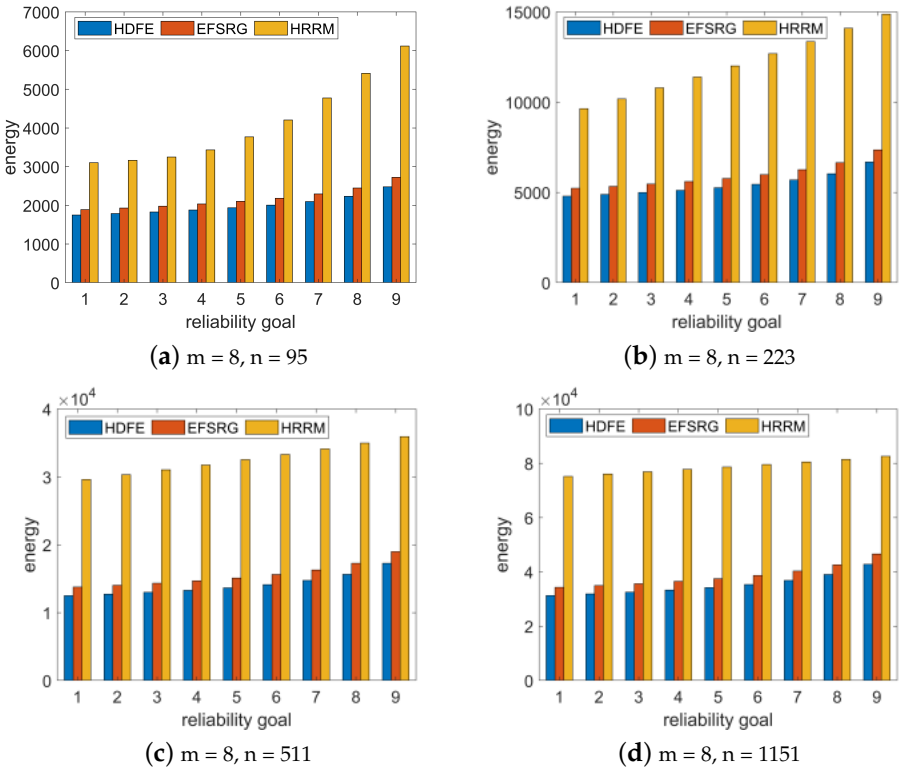



**Figure 3.** Fast Fourier Transform application when the number of size of input vectors is four.

We randomly generated the FFT application when the number of size of input vectors $N$ = 16, 32, 64, and 128 (that is, the number of tasks is 95, 223, 511, and 1151), and when the number of processor cores was set to eight. We increased the reliability requirement of the application from 0.90 to 0.99 in steps of 0.01, randomly generated 100 FFT applications, and compared the average energy consumption of various algorithms for scheduling applications with various reliability goals. Then, we scheduled the application by using the EFSRG algorithm, the HRRM algorithm, and our proposed HDFE algorithm and compared the scheduling results. The comparison index is the energy consumption required to meet the reliability requirements. The experimental data were subsequently drawn into a bar chart, as shown in Figure 4.

To study whether the number of processor cores has an impact on the experimental results, we randomly generated a Fast Fourier Transform application with size of input vectors $n$ = 4 and $N$ = 95 and compared the experimental results when the number of processors m was 4, 8, 16, and 20. The experimental results are shown in Figure 5.

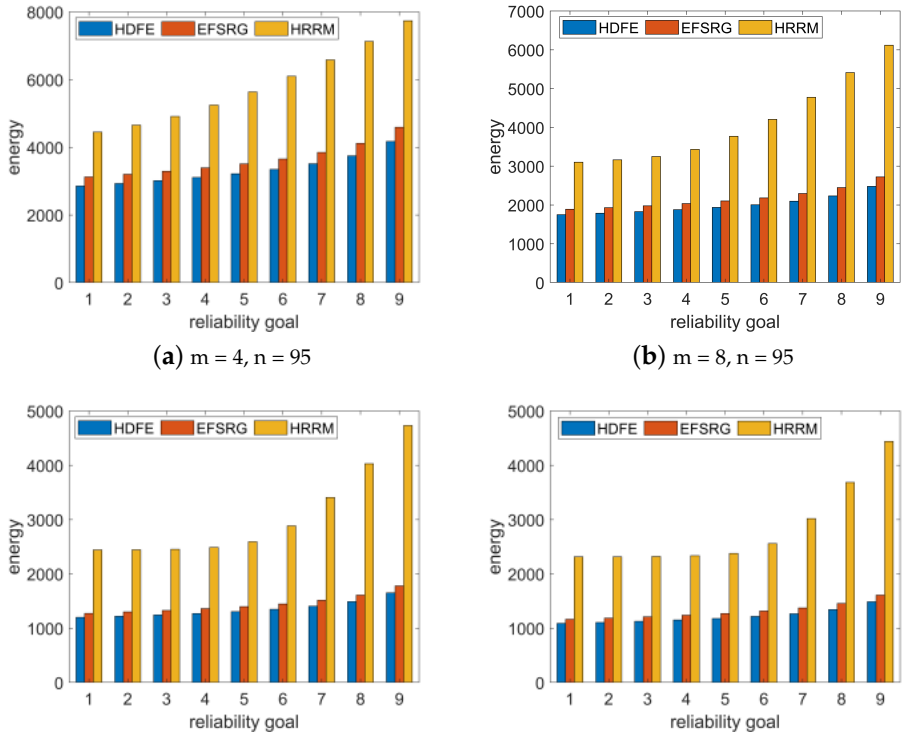
The analysis results show that in the same processor environment, the same Fast Fourier Transform application is scheduled. The HDFE algorithm proposed in this paper satisfies the required reliability requirements with a lower energy consumption than the other two algorithms, and this result is not affected by the number of task nodes and the number of processors.




(**a**) m = 8, n = 95 (**b**) m = 8, n = 223

(**c**) m = 8, n = 511 (**d**) m = 8, n = 1151

**Figure 4.** Comparative experimental results of energy consumption required by different specifications of FFT application under different reliability requirements.




(**a**) m = 4, n = 95 (**b**) m = 8, n = 95

(**c**) m = 16, n = 95 (**d**) m = 20, n = 95

**Figure 5.** Comparative experimental results of energy consumption required by the same FFT application specifications under different reliability requirements for different numbers of processor environments.

*5.2. Ge (Gaussian Elimination) Application Scheduling Experiment*

The Gaussian elimination method, an algorithm in linear algebraic programming, can be used to solve linear equations, find out the rank of the matrix, and find out the inverse matrix of the reversible equation. Figure 6 is a 5 × 5 dependency diagram of Gaussian elimination calculation based on the matrix. The relationship between the number of tasks n of the Gaussian elimination application and the number of size of input vectors n is shown in Formula (16) [23].

$$n = \left(N^2 + N - 2\right)/2,\ N \in Z^+ \tag{16}$$

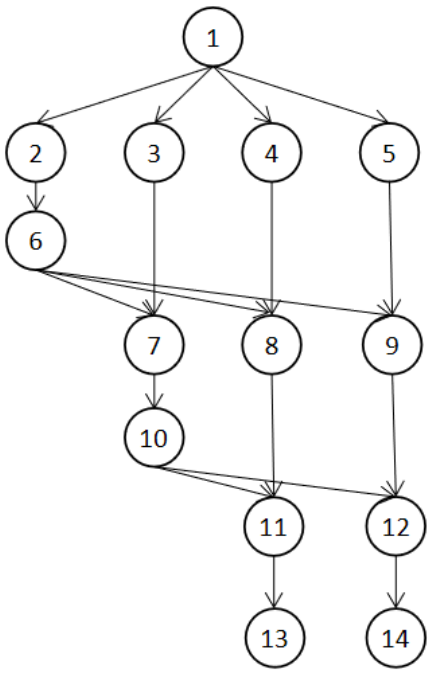
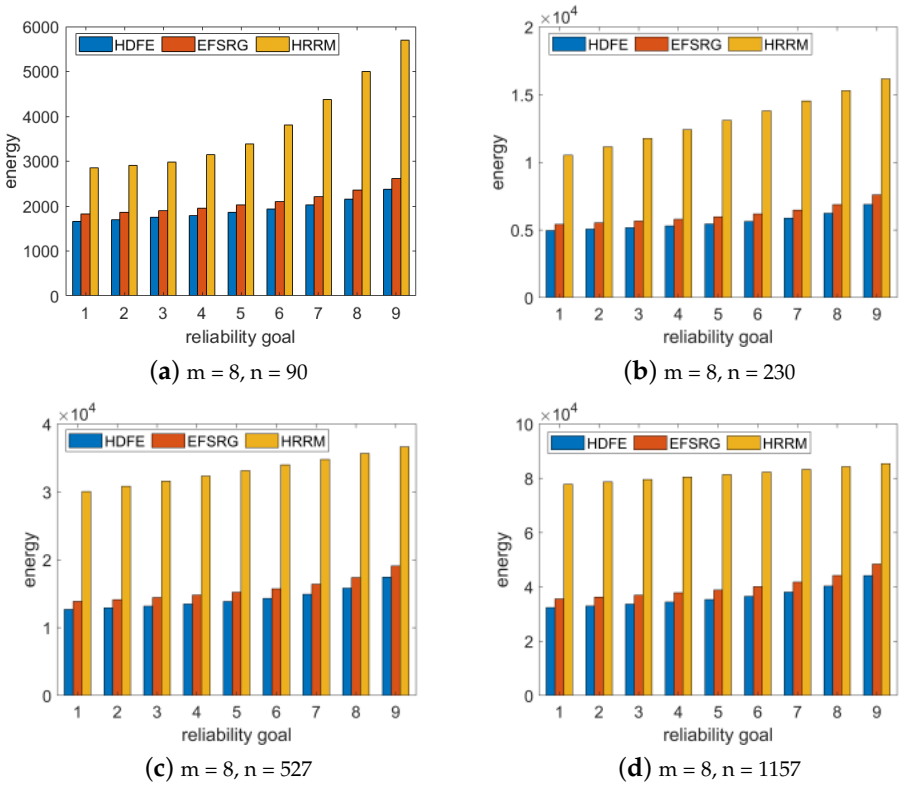



**Figure 6.** Gaussian Elimination application when the number of size of input vectors is four.

We randomly generated the Gaussian elimination application when the number of size of input vectors $n$ = 13, 21, 32, and 48 (i.e., the number of tasks is 90, 23, 05, 27, and 11, 75). We increased the reliability requirement of the application from 0.90 to 0.99 in steps of 0.01, randomly generated 100 GE applications, and compared the average energy consumption of various algorithms for scheduling applications with various reliability goals. Then, we scheduled the application by using the EFSRG algorithm, the HRRM algorithm, and our proposed HDFE algorithm and compared the scheduling results. The comparison index is the energy consumption required to meet reliability requirement. The experimental data were subsequently drawn into a bar chart, as shown in Figure 7.

To study whether the number of processor cores affects the experimental results, we randomly generated the Gaussian elimination application with matrix dimensions of $n$ = 13 and $n$ = 90 and compared the experimental results when the number of processors m was 4, 8, 16, and 20. The experimental results are shown in Figure 8.

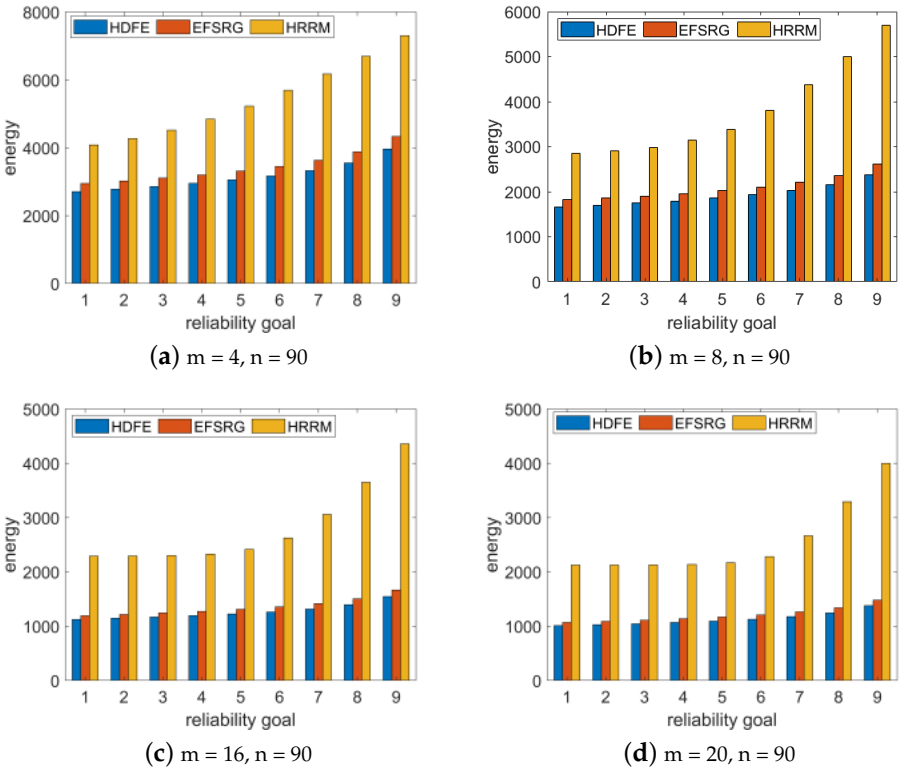
The experimental results show that the same Gaussian cancellation application is scheduled in the same processor environment, and that the energy consumption of the HDFE algorithm proposed in this paper to meet the given reliability requirements is lower than that of the other two algorithms; this result is not affected by the number of task nodes and processors.




(**a**) m = 8, n = 90 (**b**) m = 8, n = 230

(**c**) m = 8, n = 527 (**d**) m = 8, n = 1157

**Figure 7.** Comparative experimental results of energy consumption required by the GE application with different specifications under different reliability requirements.




(**a**) m = 4, n = 90 (**b**) m = 8, n = 90

(**c**) m = 16, n = 90 (**d**) m = 20, n = 90

**Figure 8.** Comparative experimental results of energy consumption required by the GE application with the same specifications under different reliability requirements for different numbers of processor environments.

## 6. Conclusions

In this paper, we tried to solve the problem of scheduling DAG applications with hard reliability requirements in heterogeneous multiprocessor systems while keeping the energy consumption as low as possible. Algorithms that adopt task replication techniques to meet the reliability requirements performed well in terms of energy savings, but task replication techniques led to a surge in overall energy consumption. In this paper, we also proposed a static heuristic scheduling algorithm, HDFE, which can achieve energy savings while satisfying the hard reliability requirements of applications. The algorithm combines task replication technology with DVFS technology based on a thorough study of the correlation between task reliability and energy consumption, it performs task replica selection based on the combined parameters of task energy consumption and reliability, and finally, it selects application replicas that meet the reliability requirements and have low energy consumption. In the experimental part, we designed a comparison experiment based on the FFT and GE applications, and the comparison algorithm we used was the EFSRG algorithm and the HRRM algorithm. Upon analysis of the experimental results, it can be concluded that the scheduling which made use of the HDFE algorithm proposed in this paper requires less energy to achieve the hard reliability requirements of the application. In future research, we will consider complicating the task model with the processor model in order to explore the task scheduling problem in complex environments.

Our proposed algorithm is for embedded, heterogeneous multi-core systems with high, soft, real-time reliability requirements. In future work, we can put emphasis on combining real-time performance, energy savings, and reliability. For example, the task deadlines will be required in the model assumptions.

**Author Contributions:** Data curation, J.W.; Resources, J.W.; Software, Y.G.; Validation, Q.C. and Y.G.; Visualization, Y.H.; Writing—original draft, Q.C.; Writing—review & editing, Q.C. and Y.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**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. Burger, D.; Goodman, J.R. Billion-transistor architectures: There and back again. *Computer* **2004**, *37*, 22–28. [CrossRef]
2. Kumar, N.; Mayank, J.; Mondal, A. Reliability aware energy optimized scheduling of non-preemptive periodic real-time tasks on heterogeneous multiprocessor system. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *31*, 871–885. [CrossRef]
3. Zhu, D.; Melhem, R.; Mossé, D. The effects of energy management on reliability in real-time embedded systems. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004, San Jose, CA, USA, 7–11 November 2004; pp. 35–40.
4. Ernst, D.; Das, S.; Lee, S.; Blaauw, D.; Austin, T.; Mudge, T.; Kim, N.S.; Flautner, K. Razor: Circuit-level correction of timing errors for low-power operation. *IEEE Micro* **2004**, *24*, 10–20. [CrossRef]
5. Xie, G.; Chen, Y.; Xiao, X.; Xu, C.; Li, R.; Li, K. Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems. *IEEE Trans. Sustain. Comput.* **2017**, *3*, 167–181. [CrossRef]
6. Xie, G.; Zeng, G.; Chen, Y.; Bai, Y.; Zhou, Z.; Li, R.; Li, K. Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems. *IEEE Trans. Serv. Comput.* **2017**, *13*, 871–886. [CrossRef]
7. Sheikh, S.Z.; Pasha, M.A. Energy-Efficient Cache-Aware Scheduling on Heterogeneous Multicore Systems. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 206–217. [CrossRef]
8. Kang, J.; Ranka, S. DVS based energy minimization algorithm for parallel machines. In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–12.
9. Devadas, V.; Aydin, H. On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications. *IEEE Trans. Comput.* **2010**, *61*, 31–44. [CrossRef]
10. Han, J.J.; Wu, X.; Zhu, D.; Jin, H.; Yang, L.T.; Gaudiot, J.L. Synchronization-aware energy management for VFI-based multicore real-time systems. *IEEE Trans. Comput.* **2012**, *61*, 1682–1696. [CrossRef]

11. Kong, F.; Yi, W.; Deng, Q. Energy-efficient scheduling of real-time tasks on cluster-based multicores. In Proceedings of the 2011 Design, Automation & Test in Europe, Grenoble, France, 14–18 March 2011; pp. 1–6.
12. Wu, X.; Lin, Y.; Han, J.J.; Gaudiot, J.L. Energy-efficient scheduling of real-time periodic tasks in multicore systems. In *IFIP International Conference on Network and Parallel Computing*; Springer: Berlin/Heidelberg, Germany, 2010, pp. 344–357.
13. Zhang, Y.W. Energy-aware mixed-criticality sporadic task scheduling algorithm. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *40*, 78–86. [CrossRef]
14. Salami, B.; Noori, H.; Naghibzadeh, M. Fairness-aware energy efficient scheduling on heterogeneous multi-core processors. *IEEE Trans. Comput.* **2020**, *70*, 72–82. [CrossRef]
15. Liu, Y.; Du, C.; Chen, J.; Du, X. Scheduling energy-conscious tasks in distributed heterogeneous computing systems. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6520. [CrossRef]
16. Iyer, R.K.; Rossetti, D.J.; Hsueh, M.C. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. Comput. Syst. (TOCS)* **1986**, *4*, 214–237. [CrossRef]
17. Castillo, X.; McConnel, S.R.; Siewiorek, D.P. Derivation and calibration of a transient error reliability model. *IEEE Trans. Comput.* **1982**, *31*, 658–671. [CrossRef]
18. Shatz, S.M.; Wang, J.P. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. *IEEE Trans. Reliab.* **1989**, *38*, 16–27. [CrossRef]
19. Zhao, B.; Aydin, H.; Zhu, D. On maximizing reliability of real-time embedded applications under hard energy constraint. *IEEE Trans. Ind. Inform.* **2010**, *6*, 316–328. [CrossRef]
20. Haque, M.A.; Aydin, H.; Zhu, D. On reliability management of energy-aware real-time systems through task replication. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *28*, 813–825. [CrossRef]
21. Roy, A.; Aydin, H.; Zhu, D. Energy-aware primary/backup scheduling of periodic real-time tasks on heterogeneous multicore systems. *Sustain. Comput. Inform. Syst.* **2021**, *29*, 100474. [CrossRef]
22. Kada, B.; Kalla, H. A fault-tolerant scheduling algorithm based on checkpointing and redundancy for distributed real-time systems. In *Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing*; IGI Global: Hershey, PA, USA 2021; pp. 770–788.
23. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 260–274. [CrossRef]