

## Accepted Manuscript

Title: Maximizing Reliability of Energy Constrained Parallel Applications on Heterogeneous Distributed Systems

Author: Xiongren Xiao Guoqi Xie Cheng Xu Chunnian Fan  
Renfa Li Keqin Li



PII: S1877-7503(17)30493-3  
DOI: <http://dx.doi.org/doi:10.1016/j.jocs.2017.05.002>  
Reference: JOCS 671

To appear in:

Received date: 1-12-2016  
Revised date: 26-4-2017  
Accepted date: 2-5-2017

Please cite this article as: Xiongren Xiao, Guoqi Xie, Cheng Xu, Chunnian Fan, Renfa Li, Keqin Li, Maximizing Reliability of Energy Constrained Parallel Applications on Heterogeneous Distributed Systems, *Journal of Computational Science* (2017), <http://dx.doi.org/10.1016/j.jocs.2017.05.002>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

# Maximizing Reliability of Energy Constrained Parallel Applications on Heterogeneous Distributed Systems

Xiongren Xiao<sup>a,b</sup>, Guoqi Xie<sup>a,b,\*</sup>, Cheng Xu<sup>a,b</sup>, Chunnian Fan<sup>c</sup>, Renfa Li<sup>a,b</sup>, Keqin Li<sup>a,d</sup>

<sup>a</sup>College of Computer Science and Electronic Engineering, Hunan University, China

<sup>b</sup>Key Laboratory for Embedded and Network Computing of Hunan Province, China

<sup>c</sup>Nanjing University of Information Science and Technology, China

<sup>d</sup>Department of Computer Science, State University of New York, New Paltz, New York, USA

---

## Abstract

Energy is one of the primary design constraints in heterogeneous distributed systems ranging from small embedded devices to large-scale data centers, where a parallel application with precedence-constrained tasks is represented by a directed acyclic graph (DAG). Dynamic voltage and frequency scaling (DVFS) has become an important energy control technology by simultaneously scaling down processor's supply voltage and frequency while tasks are running. However, recent studies show that dynamically scaling down the chip's voltage may lead to a sharp rise in transient failures of processors, thereby affecting the reliability of the system. This study solves the problem of maximizing reliability of an energy constrained parallel application on heterogeneous distributed systems based on DVFS. The problem is decomposed into two sub-problems, namely, satisfying energy constraint and maximizing reliability. The first sub-problem is solved by transferring the energy constraint of the application to that of each task, and the second sub-problem is solved by heuristically scheduling each task with maximum reliability value while satisfying its energy constraint. Experiments with real parallel applications show that the proposed MREC algorithm can obtain larger reliability values than the state-of-the-art reliability maximum energy conservation (RMEC) algorithm while satisfying the energy constraints.

*Keywords:* directed acyclic graph (DAG); dynamic voltage and frequency scaling (DVFS), energy, heterogeneous distributed systems, reliability

---

## 1. Introduction

### 1.1. Background

The trend reflected the ongoing progress in semiconductor technology allows for building fascinating, complex cloud, grid, and cluster computing systems [1, 2, 3, 4, 5, 6]. With the continuous improvement in integration and performance of system architecture, power consumption is gradually increasing and has become a major bottleneck in system design. Energy is one of the primary design constraints in heterogeneous distributed systems ranging from small embedded devices to large-scale data centers [7]. Energy has become a major issue affecting the development and use of computing systems,

---

\*Corresponding author.

*E-mail address:* xxr@hnu.edu.cn (X. Xiao), xgqman@hnu.edu.cn (G. Xie), chengxu@hnu.edu.cn (C. Xu), fcn@nuist.edu.cn (C. Fan), lirenfa@hnu.edu.cn (R. Li), lik@newpaltz.edu (K. Li)

*Phone numbers:* +8613203157161(G. Xie)

as well as the human environment. Dynamic voltage and frequency scaling (DVFS) has become an important energy control technology by simultaneously scaling down processor's supply voltage and frequency [8, 9]. However, recent studies show that dynamically scaling down the chip's voltage may lead to a sharp rise in transient failures of processors, thereby affecting the reliability of the system. A conflicting relationship exists between energy and reliability [1, 8, 10, 11]. Energy and reliability are in conflict: lower energy would result in lower reliability, and higher energy would obtain higher reliability.

As heterogeneous multi-processors continue to scale [3, 12, 13, 14, 15, 16], increasing parallel applications with precedence constrained tasks widely exist in high-performance computing systems. A parallel application with precedence constrained tasks is represented by a directed acyclic graph (DAG) where the nodes represent the tasks and the edges represent the communication messages between tasks [12, 13, 14, 15, 16]. The state-of-the-art work investigated the problem of maximizing the reliability of an energy constrained parallel application on heterogeneous distributed systems, where the reliability maximization with energy constraint (RMEC) algorithm is presented [10]. RMEC is implemented by traversing all processor and frequency combinations and selecting the combination based on the reliability maximum energy conservative function in each task assignment. However, the RMEC algorithm can be further optimized due to the following reasons:

(1) RMEC does not aim to maximize reliability. Reliability and frequency are monotonically increasing on the same processor according to the relationship function between them (Eq. (7)). As there is monotonic increase between energy-efficient frequency and energy (Eq. (4)), reliability and energy should be monotonically increasing on the same processor. RMEC just implements a tradeoff optimization between enhancing reliability and saving energy and it does not aim to maximize reliability. Moreover, even if it finds an optimal combination, this combination is not necessarily satisfying the total energy constraint of the application. If this optimal combination is skipped, then the process may not be optimal.

(2) Due to the monotonic increase of reliability and frequency, many lower frequencies are not useful in enhancing reliability; hence, these lower frequencies do not need to be verified and can be skipped in traversing.

## 1.2. Our Contributions

Our main work in this study is to maximizing the reliability of an energy constrained parallel application in heterogeneous distributed systems. The problem is divided into two sub-problems: satisfying energy constraint and maximizing reliability. Our contributions, which comparing to the state-of-the-art work [10], are summarized as follows:

(1) Satisfying energy constraint is solved by letting the unassigned tasks be preassigned to the processor with the minimum energy, and the energy constraint of the application is then transferred to that of each task, such that the heuristic algorithm with low time complexity could be used.

(2) Maximizing reliability is solved by considering that the energy of each task can be determined before the task is assigned, we select the processor and frequency combination that has the maximum reliability value while satisfying its energy constraint. Specifically, we verify from the highest to the lowest frequencies. Once we find the frequency that satisfies the energy constraint of the application, then the frequency is optimal in this processor, and the remaining frequencies can be skipped.

(3) The extensive experiments with real parallel applications is done and the results shows that not only do the actual energy values not always exceed and are close to given energy constraints, but also that maximum reliability values are generated compared with the RMEC algorithm.

The rest of this paper is organized as follows. Section 2 reviews related studies. Section 3 builds related models. Section 4 solves the presented problem. Section 5 verifies the performance of the proposed algorithm. Section 6 concludes this study.

## 2. Related work

This section mostly reviews recent related research on energy, reliability, and their relationship of a DAG-based parallel application.

DVFS-based energy-efficient design techniques have been used for parallel applications with precedence constrained tasks. In [17], the authors presented energy-conscious scheduling to implement joint minimization between schedule length and energy of a parallel application on heterogeneous distributed systems. The problem of minimizing the schedule length of an energy constrained application with precedence constrained sequential tasks [18] and precedence constrained parallel tasks (i.e., a parallel application) [19, 20] were solved. These two works were merely interested in homogeneous systems with shared memory. In [7, 21], we studied the same problem on heterogeneous distributed systems.

In general, the transient failure-free probability of a processor is subject to the Poisson distribution [22, 2, 23, 24]. Intuitively, larger reliability could cause longer schedule length of a parallel application and the problem of optimizing schedule length and reliability is considered as a typical Bi-criteria optima or Pareto optima problem [23, 25]. In [26], the authors investigated the Bi-criteria optima problem between reliability and scheduled length by merging the Bi-criteria of scheduled length and reliability into a single objective function for joint optimization of them. The method in [26] has better reliability but longer scheduled length than that in [23]. In [25], the authors implemented the bio-objective optimization of maximizing reliability and minimizing scheduled length.

Energy and reliability have a close relationship. In [27], the authors built the relationship model between energy and reliability and solved the problem of maximizing the reliability of a parallel application with deadline and energy constraints on a single-processor. On the basis of [27], the authors further solved the problem of minimizing the energy of a parallel application with deadline constraint and reliability preservation on a single-processor [28]. In [10], the authors studied the problem of maximizing the reliability of a parallel application with energy constraint on heterogeneous distributed systems. The main limitations of [10] are summarized earlier in Section 1.2. In [11], the authors studied the joint optimization between the energy and the reliability of a parallel application with deadline constraint on heterogeneous distributed systems. This study aims to solve the same problem as [10], namely, maximizing the reliability of a parallel application with energy constraint on heterogeneous distributed systems.

## 3. Models

Table 1 gives the important notations and their definitions used in this study.

Table 1: Important notations in this study.

Notation	Definition
$c_{i,j}$	Communication time between the tasks $n_i$ and $n_j$
$w_{i,k}$	Execution time of the task $n_i$ running on the processor $u_k$ with the maximum frequency
$f_{k,ee}$	Minimum energy-efficient frequency of the processor $u_k$
$E(n_i, u_k, f_{k,h})$	Energy of the task $n_i$ on the processor $u_k$ with the frequency $f_{k,h}$
$R(n_i, u_k)$	Reliability of the task $n_i$ executed on the processor $u_k$
$R(G)$	Reliability of the parallel application $G$
$SL(G)$	Schedule length of the parallel application $G$
$\lambda_{k,h}$	Failure rate per time unit of the processor $u_k$ with the frequency $f_{k,h}$
$R(n_i, u_k, f_{k,h})$	Reliability of the task $n_i$ executed on the processor $u_k$ with the frequency $f_{k,h}$
$u_{pr(i)}$	Assigned processor of the task $n_i$
$f_{pr(i),hz(i)}$	Assigned frequency of the task $n_i$ on the processor $u_{pr(i)}$
$E_{\min}(n_i)$	Minimum energy of the task $n_i$
$E_{\max}(n_i)$	Maximum energy of the task $n_i$
$E_{\min}(G)$	Minimum energy of the parallel $G$
$E_{\max}(G)$	Maximum energy of the parallel application $G$
$E_{\text{given}}(G)$	Energy constraint of the parallel application $G$
$E_{\text{given}}(n_i)$	Energy constraint of the task $n_i$

### 3.1. Application model

Let  $U = \{u_1, u_2, \dots, u_{|U|}\}$  represent a set of heterogeneous processors, where  $|U|$  represents the size of set  $U$ . Note that for any set  $X$ , this study uses  $|X|$  to denote its size. A parallel application running on processors is represented by a DAG  $G=(N, W, M, C)$  [10, 11, 29, 12, 13, 14, 16, 15, 7, 21, 30].  $N$  represents a set of nodes in  $G$ , and each node  $n_i \in N$  represents a task.  $pred(n_i)$  represents the set of the immediate predecessor tasks of  $n_i$ .  $succ(n_i)$  represents the set of the immediate successor tasks of  $n_i$ . The task which has no predecessor task is denoted as  $n_{\text{entry}}$ ; and the task which has no successor task is denoted as  $n_{\text{exit}}$ . If an application has multiple  $n_{\text{entry}}$  or multiple  $n_{\text{exit}}$  tasks, then a dummy entry or exit task with zero-weight dependencies is added to the graph.  $W$  is an  $|N| \times |U|$  matrix where  $w_{i,k}$  denotes the execution time of  $n_i$  and runs on  $u_k$  with the maximum frequency [31].  $M$  is a set of communication edges, and each edge  $m_{i,j} \in M$  represents the communication message from  $n_i$  to  $n_j$ . Accordingly,  $c_{i,j} \in C$  represents communication time of  $m_{i,j}$  if  $n_i$  and  $n_j$  are not assigned to the same processor.

Fig. 1 shows a motivating example of a DAG-based parallel application. Table 2 is a matrix of execution time with the maximum frequency in Fig. 1. The example shows ten tasks executed on three processors  $\{u_1, u_2, u_3\}$ . The weight 14 of  $n_1$  and  $u_1$  in Table 2 represents the execution time denoted by  $w_{1,1}=14$ . We can see that the same task has different execution time on different processors due to the heterogeneity of the processors [14]. The weight 18 of edge (Fig. 1) between  $n_1$  and  $n_2$  represents the communication time denoted as  $c_{1,2}$  if  $n_1$  and  $n_2$  are not assigned to the same processor.

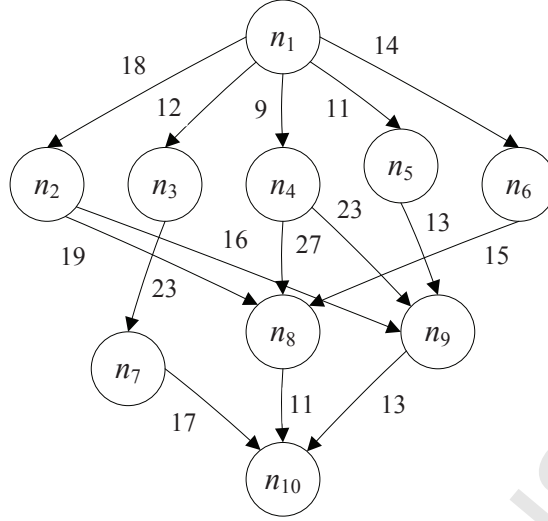


Figure 1: A motivating example of a DAG-based parallel application with ten tasks [14, 15, 7].

Table 2: Execution time of tasks on different processors with the maximum frequency of the parallel application in Fig. 1 [14, 15, 7].

Task	$u_1$	$u_2$	$u_3$
$n_1$	14	16	9
$n_2$	13	19	18
$n_3$	11	13	19
$n_4$	13	8	17
$n_5$	12	13	10
$n_6$	13	16	9
$n_7$	7	15	11
$n_8$	5	11	14
$n_9$	18	12	20
$n_{10}$	21	7	16

### 3.2. Power and energy model

Considering the almost linear relationship between the voltage and frequency, DVFS scales down the voltage alongside the frequency to save energy. Similar to [32, 27, 28, 10, 11, 7, 21], we use the term frequency change to represent changing the voltage and frequency simultaneously. Considering a DVFS-enable system, we also employ the system-level power model widely accepted in many works [32, 27, 28, 10, 11, 7, 21], where the power consumption at frequency  $f$  is calculated by

$$P(f) = P_s + h(P_{\text{ind}} + P_d) = P_s + h(P_{\text{ind}} + C_{\text{ef}}f^m). \quad (1)$$

$P_s$  represents the static power and can be removed only by powering off the whole system.  $P_{\text{ind}}$  represents frequency-independent dynamic power and can be removed by putting the system into the sleep state.  $P_d$  represents frequency-dependent dynamic power, and depends on frequencies.  $h$  represents system states and indicates whether dynamic powers are currently consumed in the system. When the system is active,  $h = 1$ ; otherwise,  $h = 0$ .  $C_{\text{ef}}$  represents effective

switching capacitance and  $m$  represents the dynamic power exponent and is no smaller than 2. Both  $C_{ef}$  and  $m$  are processor-dependent constants.

Note that there exists an excessive overhead associated with turning on/off a system,  $P_s$  is always consumed and is not manageable [32, 27, 28, 10, 11, 7, 21]. Therefore, similar to the aforementioned works, we also concentrate on managing the dynamic power, namely,  $P_{ind}$  and  $P_d$ ). Less  $P_d$  does not result in less dynamic energy due to the  $P_{ind}$  according to Eq. (1). Therefore, there is a minimum energy-efficient frequency  $f_{ee}$ , [32, 27, 28, 10, 11, 7, 21] and it is denoted by

$$f_{ee} = \sqrt[m]{\frac{P_{ind}}{(m-1)C_{ef}}}. \quad (2)$$

Assuming the frequency of a processor varies from a minimum available frequency  $f_{min}$  to the maximum frequency  $f_{max}$ , the lowest energy-efficient frequency to execute a task is

$$f_{low} = \max(f_{min}, f_{ee}). \quad (3)$$

Therefore, any actual effective frequency  $f_h$  should belong to the scope of  $f_{low} \leq f_h \leq f_{max}$ .

Considering that the number of processors is  $|U|$  in the system and these processors are completely heterogeneous, each processor should have individual power parameters [7, 21], and we define them as follows: The frequency-independent dynamic power set is

$$\{P_{1,ind}, P_{2,ind}, \dots, P_{|U|,ind}\},$$

The frequency-dependent dynamic power set is

$$\{P_{1,d}, P_{2,d}, \dots, P_{|U|,d}\},$$

The effective switching capacitance set is

$$\{C_{1,ef}, C_{2,ef}, \dots, C_{|U|,ef}\},$$

The dynamic power exponent set is

$$\{m_1, m_2, \dots, m_{|U|}\},$$

The lowest energy-efficient frequency set is

$$\{f_{1,low}, f_{2,low}, \dots, f_{|U|,low}\},$$

and the actual effective frequency set is

$$\left\{ \begin{array}{l} \{f_{1,low}, f_{1,\alpha}, f_{1,\beta}, \dots, f_{1,max}\}, \\ \{f_{2,low}, f_{2,\alpha}, f_{2,\beta}, \dots, f_{2,max}\}, \\ \dots, \\ \{f_{|U|,low}, f_{|U|,\alpha}, f_{|U|,\beta}, \dots, f_{|U|,max}\} \end{array} \right\}.$$

Finally, let  $E(n_i, u_k, f_{k,h})$  represent the dynamic energy of the task  $n_i$  on the processor  $u_k$  with frequency  $f_{k,h}$  and is calculated as

$$E(n_i, u_k, f_{k,h}) = (P_{k,ind} + C_{k,ef} \times (f_{k,h})^{m_k}) \times w_{i,k} \times \frac{f_{k,max}}{f_{k,h}}. \quad (4)$$

### 3.3. Reliability model

For a non-DVFS-enabled system, Shatz and Wang first proposed that the reliability probability for a processor is subject to the Poisson distribution [22] and it was widely accepted by numerous works [8, 27, 28, 10, 11, 22, 23, 24]. We use  $\lambda_k$  to represent the failure rate per time unit of the processor  $u_k$ , then the reliability of  $n_i$  executed on  $u_k$  in its execution time is calculated by

$$R(n_i, u_k) = e^{-\lambda_k w_{i,k}}. \quad (5)$$

Similar to most studies [27, 28, 10, 11], the reliability of the parallel application with precedence constrained tasks is denoted by

$$R(G) = \prod_{n_i \in N} R(n_i) = \prod_{n_i \in N} R(n_i, u_{pr(i)}),$$

where  $R(n_i)$  represents the actual reliability of  $n_i$  and  $u_{pr(i)}$  represents the assigned processor of  $n_i$ .

In a DVFS-capable system, different frequencies have different failure rates according to relevant research summaries [27, 28, 10, 11]. Hence, we let  $\lambda_{k,\max}$  represent the failure rate of the processor  $u_k$  with the height frequency, then the failure rate  $\lambda_{k,h}$  of the  $u_k$  with the frequency  $f_{k,h}$  is calculated as

$$\lambda_{k,h} = \lambda_{k,\max} 10^{\frac{d(f_{k,\max} - f_{k,h})}{f_{k,\max} - f_{k,\min}}}, \quad (6)$$

where  $d$  is a constant, which represents the sensitivity of failure rates to voltage scaling.

We then build the relationship between task reliability and the frequency according to Eqs. (5) and (6), namely, the reliability of the task  $n_i$  executed on the processor  $u_k$  with the frequency  $f_{k,h}$  is calculated as

$$\begin{aligned} R(n_i, u_k, f_{k,h}) &= e^{-\lambda_{k,h} \times \frac{w_{i,k} \times f_{k,\max}}{f_{k,h}}} \\ &= e^{-\lambda_{k,\max} 10^{\frac{d(f_{k,\max} - f_{k,h})}{f_{k,\max} - f_{k,\min}}} \times \frac{w_{i,k} \times f_{k,\max}}{f_{k,h}}}. \end{aligned} \quad (7)$$

We can see from Eq. (7) that the relationship between reliability and frequency is monotonically increasing on the same processor. That is, dynamically scaling down the voltage and frequency for reducing energy could result in low reliability. Then the application reliability in a DVFS-capable system is correspondingly adjusted as [27, 28, 10, 11]

$$R(G) = \prod_{n_i \in N} R(n_i, u_{pr(i)}, f_{pr(i),hz(i)}), \quad (8)$$

where  $u_{pr(i)}$  and  $f_{pr(i),hz(i)}$  represent the assigned processor and frequency of  $n_i$ , respectively,

### 3.4. Energy constraint

As the execution time of each task on each processor is known, we can get the minimum and maximum energy value denoted by  $E_{\min}(n_i)$  and  $E_{\max}(n_i)$ , respectively, by traversing all the processors [7, 21].  $E_{\min}(n_i)$  and  $E_{\max}(n_i)$  are obtained by executing the task with the maximum and minimum frequencies, respectively. Both of them are calculated by

$$E_{\min}(n_i) = \min_{u_k \in U} E(n_i, u_k, f_{k,\max}), \quad (9)$$



and

$$E_{\max}(n_i) = \max_{u_k \in U} E(n_i, u_k, f_{k,\text{low}}), \quad (10)$$

respectively.

As the energy of the application  $G$  is the sum of that of each task, we can obtain that the minimum and maximum energy value of  $G$  are

$$E_{\min}(G) = \sum_{i=1}^{|N|} E_{\min}(n_i), \quad (11)$$

and

$$E_{\max}(G) = \sum_{i=1}^{|N|} E_{\max}(n_i), \quad (12)$$

respectively.

Assume that the given energy constraint of  $G$  is  $E_{\text{given}}(G)$ , then it should be larger than or equal to  $E_{\min}(G)$ ; otherwise,  $E_{\text{given}}(G)$  is always satisfied. Meanwhile,  $E_{\text{given}}(G)$  should be less than or equal to  $E_{\max}(G)$ ; otherwise,  $E_{\text{given}}(G)$  is always not satisfied. Hence, this study assumes that  $E_{\text{given}}(G)$  belongs to the scope  $E_{\min}(G)$  and  $E_{\max}(G)$ , namely,

$$E_{\min}(G) \leq E_{\text{given}}(G) \leq E_{\max}(G).$$

### 3.5. Problem description

The problem to be addressed in this study is to assign an available processor with a proper frequency for each task, while minimizing the schedule length of the application and ensuring that the consumed energy of the application not exceed the energy constraint. The formal description is finding the processor and frequency assignments of all tasks to minimize the schedule length of the application:

$$R(G) = \prod_{n_i \in N} R(n_i, u_{pr(i)}, f_{pr(i),hz(i)}),$$

subject to its energy constraint:

$$E_{\text{total}}(G) = \sum_{i=1}^{|N|} E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leq E_{\text{given}}(G), \quad (13)$$

and  $f_{pr(i),\text{low}} \leq f_{pr(i),hz(i)} \leq f_{pr(i),\text{max}}$ , for  $\forall i : 1 \leq i \leq |N|, u_{pr(i)} \in U$ .

We know that the problem of mapping tasks to multiple multiprocessors is NP-hard [33]. Therefore, we use heuristic list scheduling to solve the subject problem in this study. List scheduling is the most well-known method for a DAG-based parallel application [12, 14, 15, 17, 7, 21], and it includes two phases. The first phase orders tasks based on the descending order of priorities (task prioritizing), whereas the second phase allocates each task to the appropriate processor (task allocation). In this study, task allocation is decomposed into two sub-problems: satisfying the energy constraint and maximizing reliability.

## 4. Proposed algorithm

### 4.1. Task prioritizing

We first need to determine the task assignment order before assigning tasks to processors. Similar to [14, 17, 7, 21], we employ the upward rank value ( $rank_u$ ) of a task given by Eq. (14) as the common task priority standard:

$$rank_u(n_i) = \bar{w}_i + \max_{n_j \in succ(n_i)} \{c_{i,j} + rank_u(n_j)\}, \quad (14)$$

where  $\bar{w}_i$  represents the average execution time of task  $n_i$  and is calculated as

$$\bar{w}_i = \left( \sum_{k=1}^{|U|} w_{i,k} \right) / |U|.$$

All the tasks are ordered according to the decreasing order of  $rank_u$ . Table 3 shows the upward rank values of all the tasks (Fig. 1). Note that only if all the predecessors of  $n_i$  have been assigned to the processors, will  $n_i$  prepare to be assigned. Assume that two tasks  $n_i$  and  $n_j$  satisfy  $rank_u(n_i) > rank_u(n_j)$ , if no precedence constraint exists between  $n_i$  and  $n_j$ ,  $n_i$  does not necessarily take precedence  $n_j$  to be assigned. We can draw that the task assignment order in  $G$  is  $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$ .

Table 3: Upward rank values of the tasks of the motivating parallel application in Fig. 1 [14].

Task	$n_1$	$n_2$	$n_3$	$n_4$	$n_5$	$n_6$	$n_7$	$n_8$	$n_9$	$n_{10}$
$rank_u(n_i)$	108	77	80	80	69	63.3	42.7	35.7	44.3	14.7

### 4.2. Satisfying energy constraint

As the task prioritization has been determined, we can then do the following work to implement task's energy constraint. Assume that the task to be assigned is  $n_{seq(j)}$ , where  $seq(j)$  represents the  $j$ th assigned task (sequence number), then  $\{n_{seq(1)}, n_{seq(2)}, \dots, n_{seq(j-1)}\}$  represents the task set where the tasks have been assigned, and  $\{n_{seq(j+1)}, n_{seq(j+2)}, \dots, n_{seq(|N|)}\}$  represents the task set where the tasks have not been assigned. To ensure that the energy constraint of the application is satisfied at each task assignment, we let each unassigned task in  $\{n_{seq(j+1)}, n_{seq(j+2)}, \dots, n_{seq(|N|)}\}$  is preassigned to the processor and frequency with the minimum energy value. Hence, when assigning  $n_{seq(j)}$ , the energy of  $G$  is calculated as

$$E_{seq(j)}(G) = \sum_{x=1}^{j-1} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x))}, hz(seq(x))) \\ + E(n_{seq(j)}, u_k, f_{k,h}) + \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}).$$

As the unassigned tasks are preassigned to the processor and frequency with the minimum energy value, any task  $n_{seq(j)}$ , only it satisfies

$$E_{seq(j)}(G) = \sum_{x=1}^{j-1} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x))}, hz(seq(x))) \\ + E(n_{seq(j)}, u_k, f_{k,h}) + \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}) \\ \leq E_{\text{given}}(G). \quad (15)$$

then the actual total energy of  $E_{\text{total}}(G)$  should be less than or equal to  $E_{\text{given}}(G)$ .

Then, we can give the energy constraint of each task before we propose the algorithm. According to Eq. (15), we have

$$\begin{aligned} E(n_{\text{seq}(j)}, u_k, f_{k,h}) &\leq E_{\text{given}}(G) \\ &- \sum_{x=1}^{j-1} E(n_{\text{seq}(x)}, u_{pr(\text{seq}(x))}, f_{pr(\text{seq}(x))}, hz(\text{seq}(x))) \\ &- \sum_{y=j+1}^{|N|} E_{\min}(n_{\text{seq}(y)}). \end{aligned}$$

Hence, let the energy constraint of the task  $n_{\text{seq}(j)}$  be

$$\begin{aligned} E_{\text{given}}(n_{\text{seq}(j)}) &= E_{\text{given}}(G) \\ &- \sum_{x=1}^{j-1} E(n_{\text{seq}(x)}, u_{pr(\text{seq}(x))}, f_{pr(\text{seq}(x))}, hz(\text{seq}(x))) \\ &- \sum_{y=j+1}^{|N|} E_{\min}(n_{\text{seq}(y)}), \end{aligned} \tag{16}$$

then, we can transfer the energy constraint of the application to that of each task. That is, we just let  $n_{\text{seq}(j)}$  satisfy the following constraint:

$$E(n_{\text{seq}(j)}, u_k, f_{k,h}) \leq E_{\text{given}}(n_{\text{seq}(j)}).$$

Hence, when assigning the task  $n_{\text{seq}(j)}$ , we can directly consider the energy constraint  $E_{\text{given}}(n_{\text{seq}(j)})$  of  $n_{\text{seq}(j)}$  and do not have to be concerned about the energy constraint of the application  $G$ . In this way, a low time complexity heuristic algorithm can be achieved. As the maximum energy constraint of  $n_{\text{seq}(j)}$  is  $E_{\max}(n_{\text{seq}(j)})$ ,  $E_{\text{given}}(n_{\text{seq}(j)})$  should be required to satisfy the following constraint:

$$E(n_{\text{seq}(j)}, u_k, f_{k,h}) \leq \min\{E_{\text{given}}(n_{\text{seq}(j)}), E_{\max}(n_{\text{seq}(j)})\}. \tag{17}$$

#### 4.3. Maximizing reliability with proposed algorithm

Inspired by the above analysis, we propose the algorithm called maximize reliability with energy constraint (MREC). The steps of MREC are described in Algorithm 1.

The main idea of MREC is that the energy constraint of the application is transferred to that of each task. Each task just selects the processor and frequency with the maximum reliability while satisfying its energy constraint. The main advantages are explained as follows.

(1) MREC has obtained the energy constraint of each task before it prepares to be assigned (Line 5).

(2) MREC skips the processor and frequency combinations that do not satisfy the energy constraint (Lines 10-12).

That is, MREC does not need to calculate the total energy value of the application and to determine whether it satisfies the given energy constraint in each task assignment by traversing all the tasks in the parallel application. On the contrary, RMEC (proposed in [10]) must do such operation.

**Algorithm 1** The MREC Algorithm**Input:**  $U = \{u_1, u_2, \dots, u_{|U|}\}$ ,  $G=(N, W, M, C)$ **Output:**  $E_{\text{total}}(G)$ ,  $R(G)$ 


---

```

1: Sort the tasks in a list downward_task_list by descending order of  $rank_u$  values.
2: while (there are tasks in downward_task_list) do
3:    $n_i = \text{downward\_task\_list.out}()$ ;
4:   Calculate  $E_{\min}(n_i)$  and  $E_{\max}(n_i)$  using Eqs. (9) and (10), respectively;
5:   Calculate  $E_{\text{given}}(n_i)$  using Eq. (16); // 1 calculate the energy constraint of  $n_i$  before it prepares to be assigned.
6:   var  $pr(i) = NULL$ ,  $f_{pr(i),hz(i)} = NULL$ ,  $R(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = 0$ ,  $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = 0$ ;
7:   for (each processor  $u_k \in U$ ) do
8:     for (each frequency  $f_{k,h}$  in from  $f_{k,\max}$  and  $f_{k,\text{low}}$ ) do
9:       Calculate  $E(n_i, u_k, f_{k,h})$  using Eq. (4);
10:      if ( $E(n_i, u_k, f_{k,h}) > \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}$ ) then
11:        continue; // 2 skip the processor and frequency combinations that do not satisfy the energy constraint of  $n_i$ .
12:      end if
13:      Calculate  $R(n_i, u_k, f_{k,h})$  using Eq. (7);
14:      if ( $R(n_i, u_k, f_{k,h}) > R(n_i, u_{pr(i)}, f_{pr(i),hz(i)})$ ) then
15:         $pr(i) = k$ ;
16:         $f_{pr(i),hz(i)} = f_{k,h}$ ;
17:         $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = E(n_i, u_k, f_{k,h})$ ;
18:         $R(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = R(n_i, u_k, f_{k,h})$ ; // 3 select the processor and frequency combination with the maximum
         $R(n_i, u_{pr(i)}, f_{pr(i),hz(i)})$ .
19:      break; // 4 skip the lower frequencies that generate lower reliability.
20:    end if
21:  end for
22: end for
23: end while
24: Calculate the actual energy value  $E_{\text{total}}(G)$  using Eq. (13);
25: Calculate  $R(G)$  using Eq. (8).

```

---

(3) MREC selects the processor and frequency combination with the maximum reliability for each task while satisfying the condition of  $E(n_i, u_k, f_{k,h}) \leq \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}$  (Lines 14-20). On the contrary, RMEC just implements a tradeoff optimization between enhancing reliability and saving energy and it does not aim to maximize reliability.

(4) As the monotone increasing relationship between reliability and frequency, once a processor and frequency combination was found, then the remaining lowest frequencies can be skipped ( Line 19). On the contrary, RMEC should traverse all the processor and frequency combinations to find an optimal RME.

The time complexity of the MREC algorithm is analyzed as follows. Scheduling all tasks must traverse all tasks, which can be done in  $O(|N|)$  time. Calculating the maximum reliability value of each task can be done in  $O(|U| \times |F|)$  time, where  $|F|$  represents the maximum number of discrete frequencies from the lowest to the maximum actual effective frequencies. Hence, the time complexity of the MRCRG algorithm is  $O(|N| \times |U| \times |F|)$ . Therefore, MREC implements low time complexity scheduling to maximize reliability of an energy constrained parallel application.

#### 4.4. Example of the MREC algorithm

This subsection gives an example to show the results using the MREC algorithm. We assume that the power parameters for all processors are known and shown in Table 4, where the maximum frequency  $f_{k,\max}$  for each processor is 1 and the frequency precision is set at 0.01.

Table 4: Power and failure parameters of processors ( $u_1$ ,  $u_2$ , and  $u_3$ ).

$u_k$	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	$m_k$	$f_{k,\text{low}}$	$f_{k,\text{max}}$	$\lambda_{k,\text{max}}$
$u_1$	0.03	0.8	2.9	0.26	1.0	0.00015
$u_2$	0.04	0.7	2.5	0.27	1.0	0.00020
$u_3$	0.07	1.0	2.5	0.29	1.0	0.00025

We can obtain the lowest energy-efficient frequency  $f_{k,\text{low}}$  for each processor according to Eq. (3). We can calculate that the minimum and maximum energy values are  $E_{\min}(G) = 19.9463$  and  $E_{\max}(G) = 157.74$  according to Eqs. (11) and (12), respectively, for the motivating parallel application. We set the energy constraint of  $G$  as  $E_{\text{given}}(G) = 3 \times E_{\min}(G) = 59.8390$ .

Table 5 shows the task assignment process of the motivating parallel application using MREC, where each row represents a task assignment and all the tasks select individual processor and frequency combinations and satisfy their individual energy constraints. We can see that only partial tasks are assigned to the processors with low frequencies (denoted with bold texts).

Table 5: Task assignment process of the motivating parallel application using MREC.

$n_i$	Task's energy constraint $E_{\text{given}}(n_i)$	Assigned processor $u_{pr(i)}$	Assigned frequency $f_{pr(i),hz(i)}$	Actual energy value $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)})$	Final reliability value $R(n_i, u_{pr(i)}, f_{pr(i),hz(i)})$
$n_1$	11.84	$u_1$	1.0	11.62	0.9979
$n_3$	20.33	$u_1$	1.0	9.13	0.9984
$n_4$	18.19	$u_2$	1.0	5.92	0.9984
$n_2$	19.26	$u_1$	1.0	10.79	0.9981
$n_5$	10.7	$u_1$	1.0	9.96	0.9982
$n_6$	5.611	$u_1$	<b>0.68</b>	5.5716	0.9799
$n_9$	2.9958	$u_2$	<b>0.3</b>	2.9803	0.9223
$n_7$	1.2563	$u_1$	<b>0.28</b>	1.2486	0.9628
$n_8$	0.8940	$u_1$	<b>0.28</b>	0.8919	0.9733
$n_{10}$	1.7266	$u_2$	<b>0.28</b>	1.7260	0.9510
$E_{\text{total}}(G) = 59.8384 \leq E_{\text{given}}(G) = 59.8390$ , $R(G) = 0.8200$					

Fig. 2 also shows the scheduling of the motivating parallel application  $G$  using MREC, where the schedule length is  $SL(G) = 147.98$ . Note that the arrows in Fig. 2 represent generated communication time between tasks. We can see that  $n_6$ ,  $n_9$ ,  $n_7$ ,  $n_8$ , and  $n_{10}$  (denoted with bold texts) are assigned to the processors with low frequencies.

Finally, the actual consumed energy of the application is  $E_{\text{total}}(G) = 59.8384$ , which is less than and close to  $E_{\text{given}}(G) = 59.8390$ . The final schedule length is  $SL(G) = 147.98$ . This example also verifies that using MREC can ensure that the

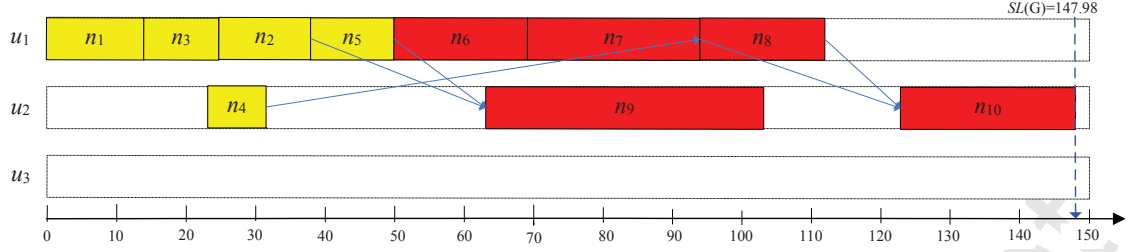


Figure 2: Scheduling of the motivating parallel application using MREC.

actual consumed energy does not exceed the given energy constraint, namely,  $E_{\text{total}}(G) \leq E_{\text{given}}(G)$ .

## 5. Experiments

### 5.1. Experimental metrics

The performance metrics selected for comparison are the actual energy value  $E_{\text{total}}(G)$  (Eq. (13)) and the final reliability  $R(G)$  (Eq. (8)) of the application. The compared algorithm with the proposed MREC algorithm is RMEC [10]. Processor and application parameters refer [27, 28] and are below:  $10 \text{ h} \leq w_{i,k} \leq 100 \text{ h}$ ,  $10 \text{ h} \leq c_{i,j} \leq 100 \text{ h}$ ,  $0.03 \leq P_{k,\text{ind}} \leq 0.07$ ,  $0.8 \leq C_{k,\text{ef}} \leq 1.2$ ,  $2.5 \leq m_k \leq 3.0$ ,  $f_{k,\text{max}} = 1 \text{ Ghz}$ , and  $0.0000001 \leq \lambda_k \leq 0.0000128$ . All frequencies are discrete, and the precision is 0.01 Ghz. All parallel applications will be executed in a simulated heterogeneous multi-processor platform with 128 processors by creating 128 process objects. Real parallel applications with precedence constrained tasks are widely used in high-performance computing, such as the GE and the FFT [14], which are two typical parallel applications with high and low parallelism, respectively. To verify effectiveness and reality, we use these two types of real parallel applications to observe the results.

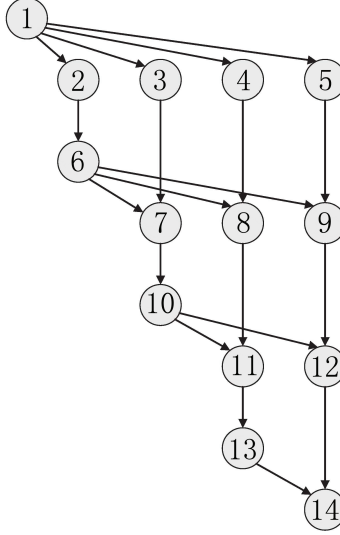
### 5.2. GE Experiments

**Experiment 1.** This experiment is conducted to compare the actual energy values and final reliability of GE applications for varying energy constraints. A new parameter  $\rho$  is used as the matrix size of the GE application, and the total number of tasks is [14]

$$|N| = \frac{\rho^2 + \rho - 2}{2}.$$

Fig. 3 shows an example of the GE parallel application with  $\rho=5$ . We limit the size of the application as  $\rho = 32$  (i.e.,  $|N| = 233$ ).  $E_{\text{given}}(G)$  is changed from  $(E_{\text{min}}(G) + E_{\text{max}}(G))/10$  to  $(E_{\text{min}}(G) + E_{\text{max}}(G))/6$ .

We can see from Table 6 that the actual energy values of the application using both RMEC and MREC can satisfy the energy constraints in all cases; however, RMEC always generates the same energy value 1306.3203, which is close to the minimum energy value  $E_{\text{min}}(G) = 1306.0969$  rather than the given energy constraints; such phenomenon reflects the facts that: 1) RMEC is not sensitive to given energy constraints; 2) RMEC just implements a tradeoff between enhancing reliability and saving energy, and it is not to maximize reliability. On the contrary, the generated energy values using MREC are less than and close to given energy constraints in all cases; the results demonstrate that the strategy (i.e., MREC transfers the energy constraint of the application to that of each task) is very effective.

Figure 3: Example of the GE application with  $\rho=5$ .Table 6: Actual energy values (GWh) and final reliability values of the GE application with  $\rho = 32$  ( $|N| = 527$ ) for varying energy constraints (**Experiment 1**).

$ N $	$E_{\min}(G)$	$E_{\max}(G)$	$E_{\text{given}}(G)$	RMEC [10]		MREC	
				$E_{\text{total}}(G)$	$R(G)$	$E_{\text{total}}(G)$	$R(G)$
527	1306.0969	64001.7080	6530.7807	1306.3203	0.1898	6530.7808	0.3934
527	1306.0969	64001.7080	7256.4230	1306.3203	0.1898	7256.4239	0.4394
527	1306.0969	64001.7080	8163.4759	1306.3203	0.1898	8163.4769	0.4932
527	1306.0969	64001.7080	9329.6867	1306.3203	0.1898	9329.6876	0.5784
527	1306.0969	64001.7080	10884.6345	1306.3203	0.1898	10884.6354	0.7113

The final reliability values using RMEC are fixed at 0.1898, whereas those using MREC are enhanced gradually from 0.3934 to 0.7113 with the increase of energy constraints, which also verify that energy and reliability are monotonically increasing in energy-efficient frequencies. In a word, MREC can obtain much higher reliability values than RMEC while satisfying the given energy constraints.

### 5.3. FFT Experiments

**Experiment 2.** To further verify the fact that MREC can obtain much higher reliability values than RMEC while satisfying the given energy constraints, we use another important real parallel application (i.e., the GE application) as experimental object. A new parameter  $\rho$  is used as the size of the FFT application, and the total number of tasks is [14]

$$|N| = (2 \times \rho - 1) + \rho \times \log_2 \rho,$$

where  $\rho = 2^y$  for some integer  $y$ . Fig. 4 shows an example of the FFT parallel application with  $\rho=8$ . Note that  $\rho$  exit tasks exist in the FFT application with the size  $\rho$ . To adapt the application model of this study, we just add a virtual exit task and the last  $\rho$  tasks are set as the immediate predecessor tasks of the virtual task. This experiment is to compare

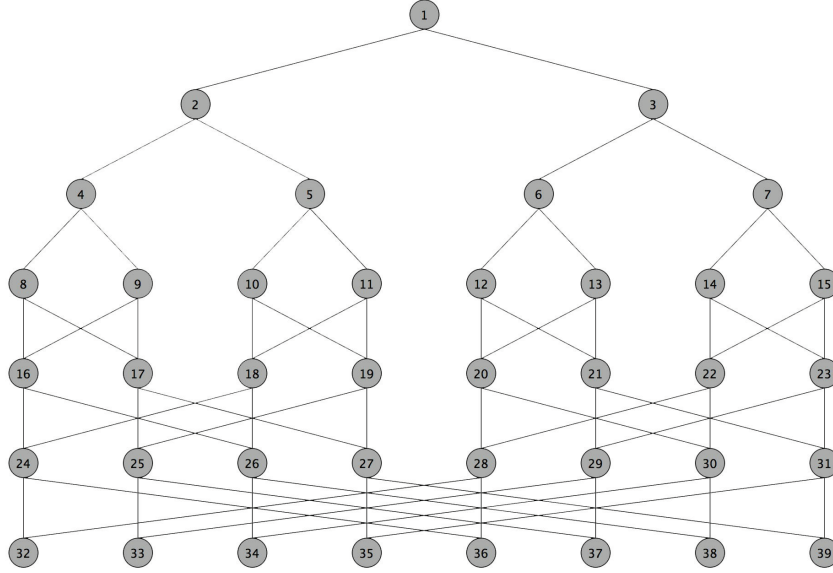


Figure 4: Example of the FFT application with  $\rho=8$ .

the actual energy values and the final reliability of the FFT application for varying energy constraints. We limit the size of the application as  $\rho = 64$ , i.e., the number of tasks are  $|N| = 512$ , which is approximately equal to that of the GE application in **Experiment 1**.  $E_{\text{given}}(G)$  is also changed from  $(E_{\text{min}}(G) + E_{\text{max}}(G))/10$  to  $(E_{\text{min}}(G) + E_{\text{max}}(G))/6$ .

Table 7: Actual energy values (GWh) and final reliability values of the FFT application with  $\rho = 64$  ( $|N| = 512$ ) for varying energy constraints (**Experiment 2**).

$ N $	$E_{\text{min}}(G)$	$E_{\text{max}}(G)$	$E_{\text{given}}(G)$	RMEC [10]		MREC	
				$E_{\text{total}}(G)$	$R(G)$	$E_{\text{total}}(G)$	$R(G)$
512	1246.0493	62388.5500	6363.4599	1246.5818	0.2513	6530.7808	0.4644
512	1246.0493	62388.5500	7070.5110	1246.5818	0.2513	7070.5121	0.5052
512	1246.0493	62388.5500	7954.3249	1246.5818	0.2513	7954.3258	0.5642
512	1246.0493	62388.5500	9090.6570	1246.5818	0.2513	9090.6582	0.6341
512	1246.0493	62388.5500	10605.7665	1246.5818	0.2513	10605.7676	0.7575

The results of Table 7 in **Experiment 2** illustrate the same patterns as that of Table 6 in **Experiment 1**: 1) RMEC still generates the same energy value 1246.5818, which is close to the minimum energy value  $E_{\text{min}}(G) = 1246.0493$  rather than the given energy constraints. 2) The generated energy values using MREC are still less than and close to the given energy constraints in all cases; 3) the final reliability values using RMEC are fixed at 0.2513, whereas those using MREC are enhanced gradually from 0.4644 to 0.7575 with the increase of energy constraints. Therefore, MREC obtains much higher reliability values than RMEC while satisfying the given energy constraints with different types of parallel applications.

**Experiment 3.** To observe the performance in different scales of applications, this experiment is conducted to compare the actual energy values and final reliability values of the FFT applications for varying numbers of tasks. We



limit  $E_{\text{given}}(G)$  as  $E_{\text{given}}(G) = (E_{\text{min}}(G) + E_{\text{max}}(G))/6$ .  $\rho$  is changed from 8 to 128, namely, the number of tasks are changed from 128 (i.e.,  $|N| = 1152$ ) to 1024 (i.e.,  $|N| = 12288$ ).

Table 8: Actual energy values (GWh) and final reliability values of FFT applications with  $E_{\text{given}}(G) = (E_{\text{min}}(G) + E_{\text{max}}(G))/6$  for varying numbers of tasks (**Experiment 3**).

$ N $	$E_{\text{min}}(G)$	$E_{\text{max}}(G)$	$E_{\text{given}}(G)$	RMEC [10]		MREC	
				$E_{\text{total}}(G)$	$R(G)$	$E_{\text{total}}(G)$	$R(G)$
1152	2893.3353	138153.49	23507.8042	2893.4961	0.0247	23507.8058	0.2660
2560	5967.0006	311557.4	52920.7334	5968.2080	$7.3906 \times 10^{-4}$	52920.7348	0.3707
5632	13037.9147	678021.5900	115176.5841	13042.1533	$5.7238 \times 10^{-8}$	115176.5858	0.0066
12288	28835.5678	1469497.7500	249722.2196	28841.1020	$3.6737 \times 10^{-16}$	249722.2211	0.0018

As can be seen from Table 8, although different scale applications are tested, the energy values generated using RMEC are still close to the minimum energy values, whereas those using MREC are close to the given energy constraints in all cases. The results also show that the reliability values using RMEC and MREC are reduced to unacceptable levels, especially using RMEC. For example, when the task number is 12288, the reliability value using RMEC is merely  $3.6737 \times 10^{-16}$ , whereas that using MREC is also reduced to 0.0018, but it is much higher than the value using RMEC.

Combined with the results of the FFT and the GE applications, the proposed MREC is very effective in reliability maximization while satisfying given energy constraints. We also find that RMEC is not to maximize reliability but more likely to make a tradeoff between energy and reliability.

**Experiment 4.** As the reliability values of the FFT application dropped to unacceptable levels in large-scale applications, we should take some measures to enhance the reliability in this situation. A feasible method is to employ the processors with lower failure rates. In this experiment, we compare the obtained reliability values of the FFT application for varying failure rates of processors. We limit  $\rho = 1024$  (i.e.,  $|N| = 12288$ ) and set  $E_{\text{given}}(G) = (E_{\text{min}}(G) + E_{\text{max}}(G))/6$ . The failure rates of processors decrease exponentially.

Table 9: Actual energy values (GWh) and final reliability values of the FFT application with  $\rho = 512$  and  $E_{\text{given}}(G) = (E_{\text{min}}(G) + E_{\text{max}}(G))/6$  for varying failure rates (**Experiment 4**).

$ N $	$E_{\text{min}}(G)$	$E_{\text{max}}(G)$	$E_{\text{given}}(G)$	RMEC [10]		MREC	
				$E_{\text{total}}(G)$	$R(G)$	$E_{\text{total}}(G)$	$R(G)$
$[10^{-8}, 128 \times 10^{-8}]$	28921.0423	1467719.61	249440.1087	28925.6223	0.03818	249441.1071	0.2568
$[10^{-9}, 128 \times 10^{-9}]$	28871.4554	1480608.6200	251580.0126	28876.5574	0.6808	251580.0938	0.8835
$[10^{-10}, 128 \times 10^{-10}]$	29771.3622	1493827.09	253933.0753	29778.2436	0.9597	253934.1732	0.9817

We can see from Table 9 that reliability values of the FFT application increase with the decrease in the failure rates. When failure rates belong to the scope of  $[10^{-10}, 32 \times 10^{-10}]$ , the reliability value using MREC reaches 0.9817. We can then draw a conclusion that reducing the failure rates of processors can effectively enhance the reliability of the parallel application. However, lower failure rates of processors would require higher design costs in practice.

## 6. Conclusions

We have developed a heuristic and low time complexity reliability maximization algorithm MREC for an energy constrained parallel application on heterogeneous distributed systems based on DVFS energy-efficient design technique. First, MREC transfers the energy constraint of the application to that of each task, thereby assigning each task to the processor with the maximum reliability while satisfying its energy constraint. Second, MREC can always generate the energy value that is less than and close to the given energy constraint of the parallel application. MREC overcomes the limitations of RMEC and can obtain larger reliability values than RMEC while satisfying the energy constraints with different types of real parallel applications in different conditions. We believe that the MREC algorithm could effectively improve a part of energy-reliability aware design for parallel applications in heterogeneous distributed environments. In our future work, we will include the timing constraint into the problem and simultaneously satisfy the timing and energy constraint to maximize the reliability for a parallel application on heterogeneous distributed systems.

## Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have helped to improve the manuscript. This work was partially supported by the National Key Research and Development Plan of China under Grant Nos. 2016YFB0200405 and 2012AA01A301-01, the National Natural Science Foundation of China with Grant Nos. 61672217, 61432005, 61379115, 61402170, 61370097, 61502162 and 61502405, the CERNET Innovation Project under Grant No. NGII20161003, and the China Postdoctoral Science Foundation under Grant No. 2016M592422.

## References

- [1] G. Xie, J. Jiang, Y. Liu, R. Li, K. Li, Minimizing energy consumption of real-time parallel applications on heterogeneous systems, *IEEE Transactions on Industrial Informatics* PP (Feb. 2017) 1–1.
- [2] G. Xie, G. Zeng, Y. Chen, Y. Bai, Z. Zhou, R. Li, K. Li, Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems, *IEEE Transactions on Services Computing* PP (Feb. 2017) 1–1.
- [3] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, N. Linge, A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment, *Security and Communication Networks* 9 (17) (2016) 4002–4012.
- [4] Y. Kong, M. Zhang, D. Ye, A belief propagation-based method for task allocation in open and dynamic cloud environments, *Knowledge-Based Systems* 115 (2017) 123–132.
- [5] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, K. Ren, A Privacy-Preserving and Copy-Deterrence Content-Based Image Retrieval Scheme in Cloud Computing, *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY* 11 (11) (2016) 2594–2608. doi:10.1109/TIFS.2016.2590944.

- [6] Z. Xia, X. Wang, X. Sun, Q. Wang, A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data, *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS* 27 (2) (2016) 340–352. doi:10.1109/TPDS.2015.2401003.
- [7] X. Xiao, G. Xie, R. Li, K. Li, minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems, in: *Proceedings of the 2016 14th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2016)*, IEEE Computer Society, 2016, pp. 1471–1476.
- [8] M. Lin, Y. Pan, L. T. Yang, M. Guo, N. Zheng, Scheduling co-design for reliability and energy in cyber-physical systems, *IEEE Transactions on Emerging Topics in Computing* 1 (2) (2013) 353–365.
- [9] A. Fanfakh, J. C. Charr, R. Couturier, A. Giersch, Optimizing the energy consumption of message passing applications with iterations executed over grids, *Journal of Computational Science* (2016) 1–14.
- [10] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, *Information Sciences* 319 (2015) 113–131.
- [11] L. Zhang, K. Li, K. Li, Y. Xu, Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems, *International Journal of Electrical Power and Energy Systems* 78 (2016) 499–512.
- [12] G. Xie, G. Zeng, L. Liu, R. Li, K. Li, High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems, *Journal of Systems Architecture* 70 (2016) 3–14.
- [13] G. Xie, G. Zeng, L. Liu, R. Li, K. Li, Mixed real-time scheduling of multiple dags-based applications on heterogeneous multi-core processors, *Microprocessors and Microsystems* 47 (2016) 93–103.
- [14] H. Topcuoglu, S. Hariri, M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, *Parallel and Distributed Systems*, *IEEE Transactions on* 13 (3) (2002) 260–274.
- [15] G. Xie, R. Li, K. Li, Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems, *Journal of Parallel and Distributed Computing* 83 (2015) 1–12.
- [16] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, L. Liu, Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds, *IEEE Transactions on Parallel and Distributed Systems* (2016) 1–14.
- [17] Y. C. Lee, A. Y. Zomaya, Energy conscious scheduling for distributed computing systems under different operating conditions, *IEEE Transactions on Parallel and Distributed Systems* 22 (8) (2011) 1374–1381.
- [18] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, *IEEE Transactions on Computers* 61 (12) (2012) 1668–1681.
- [19] K. Li, Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels, *Journal of Parallel and Distributed Computing* 95 (2016) 15–28.

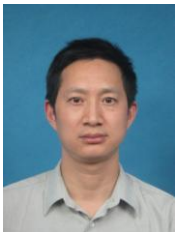
- [20] K. Li, Power and performance management for parallel computations in clouds and data centers, *Journal of Computer and System Sciences* 82 (2) (2016) 174–190.
- [21] G. Xie, X. Xiao, R. Li, K. Li, Schedule length minimization of parallel applications with energy consumption constraints using heuristics on heterogeneous distributed systems, *Concurrency and Computation: Practice and Experience* (2016) 1–10.
- [22] S. M. Shatz, J. P. Wang, Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems, *IEEE Transactions on Reliability* 38 (1) (1989) 16–27.
- [23] A. Girault, H. Kalla, A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate, *Dependable and Secure Computing*, *IEEE Transactions on* 6 (4) (2009) 241–254.
- [24] A. Benoit, L.-C. Canon, E. Jeannot, Y. Robert, Reliability of task graph schedules with transient and fail-stop failures: complexity and algorithms, *Journal of Scheduling* 15 (5) (2012) 615–627.
- [25] J. J. Dongarra, E. Jeannot, E. Saule, Z. Shi, Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems, in: *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, ACM, 2007, pp. 280–288.
- [26] A. Doğan, F. Özgüner, Biobjective scheduling algorithms for execution time–reliability trade-off in heterogeneous computing systems, *The Computer Journal* 48 (3) (2005) 300–314.
- [27] B. Zhao, H. Aydin, D. Zhu, On maximizing reliability of real-time embedded applications under hard energy constraint, *IEEE Transactions on Industrial Informatics* 6 (3) (2010) 316–328.
- [28] B. Zhao, H. Aydin, D. Zhu, Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints, *ACM Transactions on Design Automation of Electronic Systems* 18 (2) (2013) 99–109.
- [29] X. Tang, W. Tan, Energy-efficient reliability-aware scheduling algorithm on heterogeneous systems, *Scientific Programming* 2016 (2016) 1–13.
- [30] H. Arabnejad, J. G. Barbosa, Maximizing the completion rate of concurrent scientific applications under time and budget constraints, *Journal of Computational Science* (2016) 1–10.
- [31] G. Xie, L. Liu, L. Yang, R. Li, Scheduling trade-off of dynamic multiple parallel workflows on heterogeneous distributed computing systems, *Concurrency and Computation Practice and Experience* (2016) 1–18.
- [32] D. Zhu, H. Aydin, Reliability-aware energy management for periodic real-time tasks, *IEEE Transactions on Computers* 58 (10) (2009) 1382–1397.
- [33] J. D. Ullman, Np-complete scheduling problems, *Journal of Computer and System Sciences* 10 (3) (1975) 384–393.



**Xiongren Xiao** is PhD candidate and assistant professor in College of Computer Science and Electronic Engineering, Hunan University, China. His main research interests include energy-efficient computing, parallel and distributed systems. He is member of ACM and CCF.



**Guoqi Xie** received his Ph.D. degree in computer science and engineering from Hunan University, China, in 2014. He was a Postdoctoral Researcher at Nagoya University, Japan, from 2014 to 2015. Since 2015 he is working as a Postdoctoral Researcher at Hunan University, China. His major interests include parallel and distributed systems, embedded and real-time systems, software engineering and methodology. He is a member of IEEE and ACM.



**Cheng Xu** is a professor in College of Computer Science and Electronic Engineering, Hunan University, China. His major research includes embedded systems and cyber-physical systems. He is member of ACM and CCF.



**Chunnian Fan** received her Ph.D. degree in computer science from Nanjing University in 2011. Her research interests include parallel and distributed system, pattern recognition and image processing.



**Renfa Li** is a Professor of computer science and electronic engineering, and the Dean of College of Computer Science and Electronic Engineering, Hunan University, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. He is also an expert committee member of National Supercomputing Center in Changsha, China. His major interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of things. He is a member of the council of CCF, a senior member of IEEE, and a senior member of ACM.



**Keqin Li** is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 460 journal articles, book

chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing}, IEEE Transactions on Services Computing}, IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.

## Highlights

- The problem of maximizing reliability of an energy constrained parallel application on heterogeneous distributed systems is decomposed into two sub-problems satisfying energy constraint and maximizing reliability.
- The first sub-problem is solved by transferring the energy constraint of the application to that of each task.
- The second sub-problem is solved by heuristically scheduling each task with maximum reliability value while satisfying its energy constraint.