Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 9823213, 13 pages
http://dx.doi.org/10.1155/2016/9823213

*Research Article*

# Energy-Efficient Reliability-Aware Scheduling Algorithm on Heterogeneous Systems

**Xiaoyong Tang[1,2] and Weizhen Tan[3]**

[1]*School of Information Science and Engineering, National Supercomputing Center in Changsha, Hunan University, Changsha 410082, China*
[2]*Information Science and Technology College/Southern Regional Collaborative Innovation Center for Grain and Oil Crops in China, Hunan Agricultural University, Changsha 410128, China*
[3]*Archive, Hunan University of Humanities, Science and Technology, Loudi 417000, China*

Correspondence should be addressed to Weizhen Tan; twb1022@163.com

Received 22 December 2015; Accepted 24 February 2016

Academic Editor: Florin Pop



The amount of energy needed to operate high-performance computing systems increases regularly since some years at a high pace, and the energy consumption has attracted a great deal of attention. Moreover, high energy consumption inevitably contains failures and reduces system reliability. However, there has been considerably less work of simultaneous management of system performance, reliability, and energy consumption on heterogeneous systems. In this paper, we first build the precedence-constrained parallel applications and energy consumption model. Then, we deduce the relation between reliability and processor frequencies and get their parameters approximation value by least squares curve fitting method. Thirdly, we establish a task execution reliability model and formulate this reliability and energy aware scheduling problem as a linear programming. Lastly, we propose a heuristic Reliability-Energy Aware Scheduling (REAS) algorithm to solve this problem, which can get good tradeoff among system performance, reliability, and energy consumption with lower complexity. Our extensive simulation performance evaluation study clearly demonstrates the tradeoff performance of our proposed heuristic algorithm.

## 1. Introduction

For a long time, energy consumption has simply been ignored in the performance evaluation in large-scale parallel computing systems. However, Intelligence (DCDi) Industry Census reported that the amount of electricity consumed by global data centers ran up to 40 GW in 2013, and it was also with a 7% increase [1]. According to the latest world's Top 500 supercomputers Ranking, the power consumption of first supercomputer "Tianhe-2" is 17.808 MW and average power consumption for Top 10 systems in Ranking list is 6.2939 MW, respectively [2]. Thus, it is obvious that high energy cost is a key feature of designing and applying heterogeneous systems.

On the other hand, computing systems are a group of heterogeneous processors connected *via* a high-speed network that supports the execution of parallel applications. For example, the Top supercomputer "Tianhe-2" in Top 500 lists consists of Intel Xeon® E5-2692 12C 2.200 GHz and Intel Xeon Phi 31S1P (MIC) [2]. For each processor, the number of transistors integrated into today's Intel Xeon EX processor reaches to nearly 2.3 billion and its power consumption over 130 W [3]. This implies the possibility of worsening single processor reliability, eventually resulting in poorness of the whole heterogeneous system reliability. Furthermore, the modern large-scale computing systems usually have a lot of processors, such as "Tianhe-2" with 3,120,000 cores and "Titan" with 560,640 cores [2]. One of the main problems existing in this situation is system reliability, which drastically decreases as the number of processor cores increases [4]. Even when the single processor's one-hour reliability becomes very high, such as 0.999999, as the system size approaches 10,000 cores, the system's MTTF (the Mean Time to Failure) drops to less than 10 hours [4]. This also allows

us to focus primarily on the main problem of this paper, which is the *simultaneous management of system performance, reliability, and energy consumption*.

In recognition of this, we first build a reliability and energy aware task scheduling architecture including precedence-constrained parallel applications and energy consumption model on heterogeneous systems. Then, we propose the single processor failure rate model based on DVFS technique and deduce the application reliability of systems. Finally, to provide an optimum solution for this problem, we propose a heuristic Reliability-Energy Aware Scheduling (REAS) algorithm, which adopts a novel scheduling objective RE. The overall objective of this paper is trying to get good tradeoff among performance, reliability, and energy consumption.

The rest of the paper is organized as follows: the related work is summarized in Section 2. We describe the task scheduling system model in Section 3. In Section 4, we provide a system reliability model. To solve this problem, a heuristic reliability and energy aware task scheduling algorithm is proposed in Section 5. In Section 6, we verify the performance of the proposed algorithm by comparing the results obtained from performance evaluation. Finally, we summarize the contributions and make some remarks on further research in Section 7.

## 2. Related Work

The high-performance parallel application running on computing systems is usually composed of intercommunicated tasks, which are scheduled to run over different processors in the systems. In most cases, the main objective of scheduling strategies is to map the multiple interacting program tasks onto processors and order their executions so that task precedence requirements are satisfied and, in the meanwhile, the minimum schedule length (makespan) can be achieved. The problem of finding the optimal schedule is NP-complete in general [5–9]. There are many scheduling algorithms that have been proposed to deal with this problem, for example, dynamic-level scheduling (DLS) algorithm [6] and heterogeneous earliest-finish-time (HEFT) algorithm [5, 8, 10, 11].

As the energy consumption has become important issue in designing large-scale computing systems in the last few years, many techniques including dynamic voltage-frequency scaling (DVFS), dynamic powering on/off, slack reclamation, resource hibernation, and memory optimizations have been investigated and developed to reduce energy consumption [12–14]. DVFS, which is a technique in which a processor runs at a less-than-maximum frequency when it is not fully utilized in order to conserve power, is perhaps the most appealing method for reducing energy consumption [14, 15]. Most of the early DVFS-enabled researches focused on the single processor of embedded and real-time computing systems [14, 16, 17]. Recently, there has been a significant amount of work on task scheduling for heterogeneous systems using DVFS-enabled techniques. For instance, Rountree et al. focused on energy optimization of MPI program in HPC environment and proposed a linear programming (LP), which incorporates allowable time delays, communication slack, and memory pressure into its scheduling using DVFS (i.e., slack reclamation) [18]. Rizvandi et al. proposed a method to find the best frequencies of processor to obtain the optimal energy consumption [19]. Lee and Zomaya addressed the problem of scheduling precedence-constrained parallel applications on multiprocessor computer systems and their scheduling decisions are made using the relative superiority metric (RS) devised as a novel objective function [20]. In [21], Zong et al. proposed two energy-efficient scheduling algorithms (EAD and PEBD) for parallel tasks on homogeneous clusters based on duplication strategy.

All of this work demonstrated that dynamic adjusting the processor's voltage and frequency can effectively reduce system energy consumption. However, recent researches have illustrated that scaling the processor's voltage and frequency has negative impact of nanoscale semiconductor circuits's cosmic ray radiations, electromagnetic interference, and alpha particles, which enforce the unreliability of processor [22–24]. Thus, it is a good way to incorporate the reliability into energy aware scheduling based on DVFS. Recently, Zhu etc. focused on reducing energy consumption while preserving the system reliability for periodic real-time tasks [25, 26]. They proposed a reliability model that the processor's reliability decreases as scaling their voltage and frequency from max to min and incorporated the reliability requirements into heuristic energy aware task scheduling strategies. However, their techniques are not suitable for precedence-constrained parallel applications on heterogeneous systems based on DVFS-enabled processors.

Many researches had dealt with the reliability on heterogeneous systems. For example, Dogan and Özgüner introduced three reliability cost functions that were incorporated into making dynamic level (DL) and proposed a reliable dynamic level scheduling algorithm (RDLS) [27]; the goal was to minimize not only the execution time but also the failure probability of the application. In our previous work [8], we propose a scheduling algorithm which considers the task's execution reliability. Qin and Jiang investigated a dynamic and reliability-cost-driven (DRCD) scheduling algorithms for precedence-constrained tasks in heterogeneous clusters [28]. Unfortunately, those works did not consider the energy consumption and the reliability of scaling the processor's voltage and frequency. In recognition of this, we focus on the reliability and energy consumption on DVFS-enabled heterogeneous systems.

## 3. System Models

*3.1. Scheduling Architecture.* Various task scheduling architectures are proposed in literature [5, 8, 9, 14, 28, 29]. However, the energy consumption and system reliability are not effectively incorporated into scheduling. In this paper, we propose a reliability and energy aware task scheduling architecture, as depicted in Figure 1(a). It is assumed that all parallel applications, along with information provided by user, are submitted to system by a special user command. First, the parallel applications are divided as a task DAG by *Task DAG Model*. Then, the

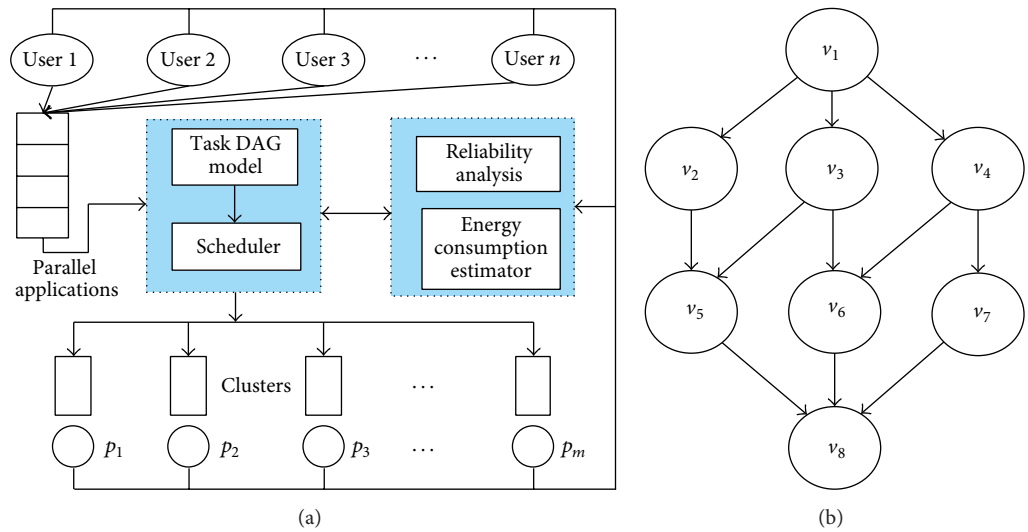


Figure 1: (a) The reliability and energy aware task scheduling architecture. (b) A parallel application task graph.

Table 1: The parameters of heterogeneous processors.

| | $\lambda_k$ | $P_s[k]$ | $\alpha[k]$ | $(f_{k,l}, V_{k,l})$ | | |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| $p_1$ | $1.4 \times 10^{-4}$ | 73.6 | $3.663 \times 10^{-8}$ | $(0.8 \times 10^9, 0.93)$ | $(2.1 \times 10^9, 1.23)$ | $(3.2 \times 10^9, 1.43)$ |
| $p_2$ | $1.62 \times 10^{-4}$ | 57.1 | $4.95 \times 10^{-8}$ | $(2.3 \times 10^9, 0.85)$ | $(3.0 \times 10^9, 1.36)$ | |

estimate energy consumption of tasks, which are executed on the DVFS-enabled heterogonous processors, is computed by the *Eneregy Consumption Estimator*. At the same time, *reliability analysis* computes the processors' reliability according to different frequency to get the whole system reliability. Finally, the *Scheduler* schedules tasks based on the above task energy consumption and system reliability.

*3.2. Heterogeneous Systems.* The target system used in this work consists of a set of $P = \{p_1, p_2, \ldots, p_m\}$ heterogeneous processors/machines [5, 8, 9, 14, 29], which are connected by high-speed interconnects, such as Infiniband and Myrinet. Each DVFS-enabled processor $p_k \in P$ can adjust its operational voltage and frequency [14]. Therefore, they can be executed on discrete set of frequency-voltage pairs, $(f_{k,l}, V_{k,l})$, in which $(f_{k,1} < f_{k,2} < \cdots < f_{k,M_k})$ and $(V_{k,1}, V_{k,2} < \cdots < V_{k,M_k})$, where $M_k$ is processor $p_k$'s operation level [14, 30]. For example, the quad-core AMD Phenom II supports 4 different frequencies (0.8 GHz, 2.1 GHz, 2.5 GHz, and 3.2 GHz) and voltages ranging from 0.85 V to 1.425 V [30]. Since clock frequency transition overheads take a negligible amount of time (e.g., 10 us–150 us), these overheads are not considered in our study.

The heterogeneous processor's failure is assumed to follow a Poisson process and each processor has a constant failure rate $\lambda$ [8, 9, 29]. For example, $\lambda_k$ denotes a processor $p_k$ failure rate when it works at normal voltage and frequency [8, 9, 27, 29]. These failure rates can be derived from system's profiling, system log, and statistical prediction techniques [31]. For demonstration purposes, we illustrate two heterogeneous processors, one has 3 frequency levels and the other has 2 frequency levels, and the parameters are listed in Table 1.

*3.3. Applications Model.* The precedence-constrained tasks of parallel application are usually denoted as a Directed Acyclic Graph (DAG) $G = \langle V, R, [d_{i,j}, w_{i,k,l}] \rangle$ [5, 8–10, 29], where $V = \{v_1, v_2, \ldots, v_n\}$ is the set with $n$ tasks that can be scheduled to any available DVFS-enabled processors [5, 8–10, 29]; $R$ represents the precedence relation that defines a partial order on the task set $V$, such that $v_i R v_j$ implies that the task $v_i$ must be finished, before $v_j$ can start execution [5, 8–10, 29]. $[d_{i,j}]$ is $n \times n$ communication matrix that denotes the communication time between tasks $v_i$ and $v_j$ for $1 \leq i, j \leq n$. $[w_{i,k,l}]$ is $n \times m \times M_{\max}$ computation matrix in which each $w_{i,k,l}$ gives the estimated time to execute task $v_i$ on processor $p_k$ at frequency $f_{k,l}$. Here, $M_{\max}$ is the maximal operation level on systems. The communication cost and computation cost can be evaluated by building a historic table and using code profiling or statistical prediction techniques [31]. Figure 1(b) shows a parallel application DAG, Table 2 lists the tasks execution time on two heterogeneous DVFS-enabled processors listed in Table 1, and the communication time among these tasks is listed in Table 3.

Generally, the common objective of task scheduling is to map tasks with precedence constrained onto processors and get a minimum schedule length (which is also called makespan) [10, 11]. Before presenting the schedule length, it is necessary to define the scheduling attributes EST and EFT

Table 2: Task estimated execution matrix $[w_{i,k,l}]$.

| Task | $p_1$ | | | $p_2$ | |
|---|---|---|---|---|---|
| | $p_{1,1}$ | $p_{1,2}$ | $p_{1,3}$ | $p_{2,1}$ | $p_{2,2}$ |
| $v_1$ | 11.12 | 2.28 | 2.87 | 3.89 | 3 |
| $v_2$ | 36.29 | 13.82 | 9.12 | 12.7 | 9.78 |
| $v_3$ | 15.46 | 5.91 | 3.92 | 5.45 | 4.21 |
| $v_4$ | 5.33 | 2.01 | 1.4 | 1.94 | 1.49 |
| $v_5$ | 66.77 | 25.44 | 16.77 | 23.28 | 17.83 |
| $v_6$ | 13.82 | 5.3 | 3.53 | 4.84 | 3.75 |
| $v_7$ | 7.43 | 2.86 | 1.89 | 2.68 | 2.04 |
| $v_8$ | 8.48 | 3.19 | 2.09 | 2.91 | 2.31 |

Table 3: Estimated communication matrix $[d_{i,j}]$.

| Task | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|
| $v_1$ | 6.99 | 15.48 | 6.69 | | | | |
| $v_2$ | | | | 10.86 | | | |
| $v_3$ | | | | 1.25 | 12.56 | | |
| $v_4$ | | | | | 6.93 | 0.3 | |
| $v_5$ | | | | | | | 0.11 |
| $v_6$ | | | | | | | 6.535 |
| $v_7$ | | | | | | | 6.2 |

of task $v_i$. $\mathrm{EST}(v_i, f_{k,l})$ denotes the earliest execution starting time of task $v_i \in V$ on DVFS-enabled processor $p_k \in P$ at frequency $f_{k,l}$, which is constrained by tasks precedence relation and the available time of processor $p_k$ [5, 8–10, 29]. $\mathrm{EFT}(v_i, f_{k,l})$ is the earliest execution finish time of task $v_i$ on processor $p_k$ at frequency $f_{k,l}$, which is described as

$$\mathrm{EFT}\left(v_i, f_{k,l}\right) = \mathrm{EST}\left(v_i, f_{k,l}\right) + w_{i,k,l}. \tag{1}$$

In this paper, let $X_{i,k}^{l} = 1$ denote the task $v_i$ scheduled on processor $p_k$ at frequency $f_{k,l}$; otherwise $X_{i,k}^{l} = 0$. Thus, the schedule length is defined as follows:

$$\text{makespan} = \max_{1\le i\le n,\, 1\le k\le m}^{1\le l\le M_k} \left\{X_{i,k}^{l}\mathrm{EFT}\left(v_i, f_{k,l}\right)\right\}. \tag{2}$$

*3.4. Energy Model.* The major energy consumption of computing systems depends on its memory, disks, CPUs, and other components. This paper only considers DVFS-enabled CPUs, which consume the largest proportion of energy on systems [14, 19, 20, 32]. The power consumption of DVFS-enabled microprocessor based on complementary metal-oxide semiconductor (CMOS) logic circuits mainly consists of *static power* and *dynamic power dissipation*, which can be modeled as [25, 26]

$$P = P_s + \emptyset P_d, \tag{3}$$

where $P_s$ is the *static power*, which is a constant and the power used to maintain basic circuits and keep the clock running, and *frequency-independent active power*. $\emptyset$ denotes the processor's model, if processor is at *execution* model, $\emptyset = 1$; otherwise, $\emptyset = 0$. $P_d$ is the most significant factor of processor power consumption and can be estimated as [14, 16, 19, 20, 32]

$$P_d = \alpha V^2 f, \tag{4}$$

where $\alpha$ represents the switched capacitance, $V$ is the supply voltage, $f$ represents processor's working frequency, and $\sigma$ stands for circuit dependent constant. The example of such processor parameters is listed in Table 1.

Let $\mathrm{EN}(v_i, f_{k,l})$ be the energy consumption caused by task $v_i$ running on DVFS-enabled processor $p_k$ at frequency $f_{k,l}$, of which it is determined by task execution time and processor power consumption:

$$\begin{aligned}\mathrm{EN}\left(v_i, f_{k,l}\right) &= w_{i,k,l} \times P^k \\ &= w_{i,k,l} \times P_s^k + w_{i,k,l} \times P_d\left(f_{k,l}\right),\end{aligned} \tag{5}$$

where $P_d(f_{k,l})$ denotes *dynamic power dissipation* of processor $p_k$ at frequency $f_{k,l}$ (see (4)). Thus, for an application $G$, the energy consumption $\mathrm{EN}(V)$ is the summation of all tasks of energy consumption:

$$\begin{aligned}\mathrm{EN}\left(V\right) &= \sum_{1\le i\le n,\, 1\le k\le m}^{1\le l\le M_k} \left\{X_{i,k}^{l}\mathrm{EN}\left(v_i, f_{k,l}\right)\right\} \\ &= \sum_{1\le i\le n,\, 1\le k\le m}^{1\le l\le M_k} \left\{X_{i,k}^{l} w_{i,k,l} \times P_s^k + X_{i,k}^{l} w_{i,k,l} \times P_d\left(f_{k,l}\right)\right\}.\end{aligned} \tag{6}$$

At the same time, for heterogeneous systems, all processors are power-on; they are sleep or execution model. That is to say, all processors of systems consume *static power* all the time. Thus, the computing systems energy consumption $\mathrm{EN}(P)$ is the summation of all processors *static power* and *dynamic power dissipation* of application energy consumption:

$$\begin{aligned}\mathrm{EN}\left(P\right) = \text{makespan} \times \sum_{k=1,2,\ldots,m} P_s^k \\ + \sum_{1\le i\le n,\, 1\le k\le m}^{1\le l\le M_k} \left\{X_{i,k}^{l} w_{i,k,l} \times P_d\left(f_{k,l}\right)\right\}.\end{aligned} \tag{7}$$

Obviously, systems energy consumption $\mathrm{EN}(P)$ is greater than application energy consumption $\mathrm{EN}(V)$. In this paper, one of our main objectives is to minimize systems energy consumption $\mathrm{EN}(P)$.

## 4. System Reliability Analysis and Problem Statement

In this section, we first provide the single DVFS-enabled processor failure rate model. Then, we analyze heterogeneous systems reliability. At last, we formulate the reliability and energy aware task scheduling as a linear programming problem.

*4.1. Single DVFS-Enabled Processor Failure Rate.* Among various sources of unreliability in a semiconductor circuit processor, it is predicted that the failure rate due to cosmic ray radiation-induced soft errors dominates all other reliability issues [24]. Transient fault occurs when a high energy particle such as alpha or neutron strikes a sensitive region in a semiconductor device and flips the logical state of the struck node [33]. Most of the modern DVFS-enabled processor is the integration of multibillion transistors on a single chip leading to increasing number of sensitive devices in submicron technologies which is vulnerable to soft error and consequently raises the *Soft Error Rate* (SER) [34]. These phenomena become more and more serious with the continued scaling of processor's voltage and frequency [23, 25].

Traditionally, the modern DVFS-enabled processor's reliability has been modeled as the following Poisson distribution with a failure rate $\lambda$ when it works at normal voltage and frequency [8, 9, 27, 29, 35]. Moreover, it has been shown that DVFS has a direct and negative effect on failure rates as blindly applying DVFS to scale the supply voltage and processing frequency for energy savings, which may cause significant degradation in processor's reliability [23, 25, 26]. Therefore, for the DVFS-enabled heterogonous processor $p_k \in P$ to be considered in this paper, the failure rate at a reduced frequency $f_{k,l}$ (and the corresponding voltage $V_{k,l}$) can be modeled as

$$\lambda_k(f_{k,l}) = \lambda_k \cdot H_k(f_{k,l}), \tag{8}$$

where $\lambda_k$ is the failure rate corresponding to the normal processing frequency $f_{\mathrm{nm}}$ (and corresponding to normal voltage $V_{\mathrm{nm}}$). Prior researches which studied the effect of normal voltage on processor's reliability have revealed that the failure rates generally increase with scaled processing frequencies (and supply voltages) away from normal voltage [24, 36]. On the other hand, the fault rates are exponentially related to the circuit's critical charge (which is the threshold voltage). Thus, we have the following equations:

$$H_k(f_{k,l}) = \begin{cases} e^{\psi_k V t_k} 10^{\xi_k((f_{k,l}-f_{\mathrm{nm}})/(f_{\max}-f_{\min}))} & f_{\mathrm{nm}} \le f_{k,l} \le f_{\max} \\ e^{\psi_k V t_k} 10^{\xi_k((f_{\mathrm{nm}}-f_{k,l})/(f_{\max}-f_{\min}))} & f_{\min} \le f_{k,l} \le f_{\mathrm{nm}}, \end{cases} \tag{9}$$

where the exponent $\psi_k$ is the parameter of threshold voltage and $\xi_k$ is a constant, representing the sensitivity of fault rates to frequency scaling, and $f_{\min}$ and $f_{\max}$ denote the minimum and maximum frequency, respectively.

In order to get the precision value of parameters $\psi_k$ and $\xi_k$, we use least squares curve fitting method [37]. Therefore, the natural logarithm of both sides for (9) is

$$\ln(H_k(f_{k,l})) = \begin{cases} \psi_k V t_k + \xi_k \ln 10 \dfrac{f_{k,l} - f_{\mathrm{nm}}}{f_{\max} - f_{\min}} & f_{\mathrm{nm}} \le f_{k,l} \le f_{\max} \\ \psi_k V t_k + \xi_k \ln 10 \dfrac{f_{\mathrm{nm}} - f_{k,l}}{f_{\max} - f_{\min}} & f_{\min} \le f_{k,l} \le f_{\mathrm{nm}}. \end{cases} \tag{10}$$

Let $y = \ln(H_k(f_{k,l}))$, $A = \psi_k V t_k$, $B = \xi_k \ln 10(1/(f_{\max} - f_{\min}))$, and $C = \xi_k \ln 10(f_{\mathrm{nm}}/(f_{\max} - f_{\min}))$. Then, (10) becomes

$$y = \begin{cases} A + B f_{k,l} - C & f_{\mathrm{nm}} \le f_{k,l} \le f_{\max} \\ A - B f_{k,l} + C & f_{\min} \le f_{k,l} \le f_{\mathrm{nm}}. \end{cases} \tag{11}$$

Thus, we can get the parameters $\psi_k$ and $\xi_k$ approximation value by using least squares linear fitting method.

*4.2. Application Reliability Analysis.* Assume that the task $v$ processing time has taken place during the time interval $[A, B]$ on heterogeneous DVFS-enabled processor $p_k$ at frequency $f_{k,l}$, where $A$ denotes the task start execution time and $B$ denotes the task finish time [5, 8, 9, 29]. Thus, the task execution reliability can be given by

$$\begin{aligned} P[v] &= P\{X(B) - X(A) = 0\} \\ &= P\{X(B - A + A) - X(A) = 0\} \\ &= \exp\{-\lambda_k(f_{k,l})(B - A)\}. \end{aligned} \tag{12}$$

For a task $v_i$ of application $G$ on processor $p_k$ at frequency $f_{k,l}$, its reliability $P[v_i, f_{k,l}]$ is equal to all of its immediate parent tasks and its execution reliability, which can be defined by

$$P[v_i, f_{k,l}] = \prod_{v_j \in \mathrm{pred}(v_i)} P[v_j] \times \exp(-\lambda_k(f_{k,l}) \times w_{i,k,l}), \tag{13}$$

where $\mathrm{pred}(v_i)$ denotes all direct predecessors of $v_i$ and $P[v_j]$ is the reliability of task $v_j$ that is equal to the reliability of task $v_i$ executing on processor $p_k$ at frequency $f_{k,l}$

$$P[v_j] = \sum_{1 \le k \le m}^{1 \le l \le M_k} \{X_{1,k}^l P[v_j, f_{k,l}]\}. \tag{14}$$

For the entry task $v_1$ of application, which is executed on processor $p_k$ at frequency $f_{k,l}$ and $\mathrm{pred}(v_1) = \phi$, its reliability

$$P[v_1, f_{k,l}] = \exp\left(-\sum_{1 \le k \le m}^{1 \le l \le M_k} \{X_{1,k}^l \lambda_k(f_{k,l}) \times w_{1,k,l}\}\right). \tag{15}$$

Generally, application $G$ has one exit task $v_{\mathrm{exit}}$. The reliability of application $P[G]$ is equal to the exit task $v_{\mathrm{exit}}$:

$$P[G] = P[v_{\mathrm{exit}}] = \prod_{v_j \in \mathrm{pred}(v_{\mathrm{exit}})} P[v_j] \times P[v_{\mathrm{exit}}, f_{k,l}]. \tag{16}$$

This is the other objective of this paper, in which we try to improve the application reliability $P[G]$. From the above analysis, we know that allocating tasks with less execution times to more reliable processors might be a good heuristic to increase the reliability.

**Input**: The task DAG of parallel applications
**Output**: The scheduling of task-processor pairs
(1) Calculate each task $b$_level of DAG
(2) Sort tasks in a scheduling list by non-increasing order of $b$_level
(3) **while** *the scheduling list isnot empty* **do**
(4) Remove the first task $v_i$ from the scheduling list
(5) Set min $F(v_i)$, min $E(v_i)$ as maximum value
(6) **for** *each processor-frequency* $f_{k,l}$ *in systems* **do**
(7) Compute the earliest finish time EFT($v_i$, $f_{k,l}$) use (22)
(8) **if** min $F(v_i)$ > EFT($v_i$, $f_{k,l}$) **then**
(9) min $F(v_i)$ = EFT($v_i$, $f_{k,l}$)
(10) **end**
(11) Compute task energy consumption EN($v_i$, $f_{k,l}$) use (5)
(12) **if** min $E(v_i)$ > EN($v_i$, $f_{k,l}$) **then**
(13) min $E(v_i)$ = EN($v_i$, $f_{k,l}$)
(14) **end**
(15) **end**
(16) Set min RE($v_i$) as maximum value
(17) **for** *each processor-frequency* $f_{k,l}$ *in systems* **do**
(18) Compute the earliest finish time EFT($v_i$, $f_{k,l}$) use (22)
(19) Compute task energy consumption EN($v_i$, $f_{k,l}$) use (5)
(20) Compute metric RE($v_i$, $f_{k,l}$) use (24)
(21) **if** min RE($v_i$) > RE($v_i$, $f_{k,l}$) **then**
(22) min RE($v_i$) = RE($v_i$, $f_{k,l}$)
(23) **end**
(24) **end**
(25) Assign task $v_i$ to the corresponding processor-frequency
(26) Update the processor execution finish time
(27) **end**
(28) "Slack reclamation
(29) **for** *each task in scheduling task-processor pairs* **do**
(30) Compute task slack time Slack($v_i$) use (25)
(31) **for** *each frequency of processor* $k$ **do**
(32) Compute the optimal frequency $f_{k,l}$ use (26)
(33) **end**
(34) Reassign task $v_i$ and update corresponding data
(35) **end**
(36) Compute the schedule length, application reliability $P[G]$, systems energy consumption EN($P$)

Algorithm 1: The pseudocode for REAS algorithm.

*4.3. Problem Statement.* As simultaneous management of scheduling performance, system reliability, and energy consumption is the main problem of this paper, we formulate it as follows:

$$\begin{aligned} &\text{Minimize} && \text{makespan} \\ &\text{Minimize} && \text{EN}(P) \\ &\text{Maximize} && P[G] \\ &\text{s.t.} && X_{i,k}^{l} = 1 \text{ Or } X_{i,k}^{l} = 0 \\ & && \sum_{1\le k\le m}^{1\le l\le M_k} X_{i,k}^{l} = 1 \quad \forall v_i \in V \\ & && v_i R v_j \quad \forall v_i, v_j \in V. \end{aligned} \tag{17}$$

## 5. Proposed Reliability-Energy Aware Scheduling Algorithm

This section presents a Reliability-Energy Aware Scheduling algorithm on heterogeneous systems called REAS, which aims at achieving lower energy consumption, high reliability, and shorter schedule length. Its scheduling decisions are made using the hybrid metric including energy consumption, reliability, and schedule length, devised as a novel objective function. The pseudocode of the algorithm is shown in Algorithm 1. The algorithm is complete in three main phases as described in the following sections.

*5.1. Task Priorities Phase.* This step is essential for list scheduling algorithms. A task processing list is generated by sorting the task by decreasing order of some predefined rank

Table 4: The $b$_level value of task.

| Task | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|
| $b$_level | 73.4 | 61.4 | 42.4 | 26 | 34.15 | 16.6 | 13.7 | 3.8 |
| Seq | 1 | 2 | 3 | 5 | 4 | 6 | 7 | 8 |

function, such as $t$_level, $b$_level, Rank, CP, and DL [5, 6, 8–10, 29]. Here, we use the average computation capacity, which is defined as

$$\overline{w(v_i)} = \frac{\sum_{1\le k\le m}^{1\le l\le M_k} w_{i,k,l}}{\sum_{1\le k\le m} M_k}. \tag{18}$$

In this research, we use $b$_level as the rank function. The $b$_level of task $v_i$ is the sum of the path weight from task $v_i$ to exit task. We can compute this value recursively traversing DAG from exit task, and it is defined as follows:

$$b\_\text{level}(v_i) = \overline{w(v_i)} + \max_{v_j\in \text{succ}(v_i)} \left\{d_{i,j} + b\_\text{level}(v_j)\right\} + \text{RC}(v_i), \tag{19}$$

where $\text{succ}(v_i)$ is the set of immediate successors of task $v_i$. $\text{RC}(v_i)$ is the average reliability overhead of task $v_i$ and can be computed by

$$\text{RC}(v_i) = \left(1 - \exp\left\{-\frac{\sum_{1\le k\le m}^{1\le l\le M_k} \lambda_k(f_{k,l})}{\sum_{1\le k\le m} M_k} \times \overline{w(v_i)}\right\}\right) \times \overline{w(v_i)}. \tag{20}$$

For the exit task $v_{\text{exit}}$, the $b$_level is equal to

$$b\_\text{level}(v_{\text{exit}}) = \overline{w(v_{\text{exit}})} + \text{RC}(v_{\text{exit}}). \tag{21}$$

Basically, $b\_\text{level}(v_i)$ is the length of the critical path from task $v_i$ to the exit task, including the average computation cost and reliability overhead of task $v_i$. For example, considering the application DAG in Figure 1(b), heterogeneous systems parameters in Table 1, task execution time matrix in Table 2, and communication matrix in Table 3, the task $b$_level value which is recursively computed by (19) and (21) is shown in Table 4.

*5.2. Task Assignment Phase.* In this phase, tasks are assigned to the processors with earliest execution finish time $\text{EFT}(v_i)$, high reliability, and minimum task energy consumption $\text{EN}(v_i)$. However, for heterogeneous systems, these performance metrics are conflicted most of the time. Here, we introduce a novel objective as RE, which can get good tradeoff among these metrics. We first redefine task $v_i$ earliest execution finish time on processor $p_k$ at frequency $f_{k,l}$ as

$$\text{EFT}(v_i, f_{k,l}) = \text{EST}(v_i, f_{k,l}) + w_{i,k,l} + \text{RO}(v_i, f_{k,l}), \tag{22}$$

where $\text{RO}(v_i, f_{k,l})$ is the reliability overhead of task $v_i$ on processor $p_k$ at frequency $f_{k,l}$ and is computed by

$$\text{RO}(v_i, f_{k,l}) = (1 - P[v_i, f_{k,l}]) \times w_{i,k,l}. \tag{23}$$

On the other hand, we let $\text{Min}\,F(v_i)$, $\text{Min}\,E(v_i)$ denote the earliest execution finish time and minimum task energy consumption on all processors of heterogeneous systems. Thus, the novel metric RE of task $v_i$ on processor $p_k$ at frequency $f_{k,l}$ is

$$\text{RE}(v_i, f_{k,l}) = \theta \times \frac{\text{EFT}(v_i, f_{k,l}) - \text{Min}\,F(v_i)}{\text{EFT}(v_i, f_{k,l})} + (1-\theta) \times \frac{\text{EN}(v_i, f_{k,l}) - \text{Min}\,E(v_i)}{\text{EN}(v_i, f_{k,l})}, \tag{24}$$

where $\theta$ is the weight of task earliest execution finish time. If the task execution time is more important than energy consumption, we can give higher value to $\theta$; otherwise, $\theta$ value is lower. Moreover, the scheduling objective of this problem is minimum in both schedule length and energy consumption. Thus, in each task assignment step, we try to get the minimum $\text{RE}(v_i, f_{k,l})$ and assign task $v_i$ to the corresponding processor frequency.

*5.3. Slack Reclamation.* Tasks of parallel application may have some slack time for their execution due primarily to communication events, for example, "multidimensional" intertask communication (or intertask data dependencies), and these processor slacks are an obvious source of energy wastage. Slack reclamation was studied to reduce energy consumption using the slack left by some completed task instances. The idea behind the slack reclamation for the reducing of energy consumption is to exploit the slack time to slow down the execution speeds of the remaining tasks [12, 20]. In this paper, we adopt this technique to reduce energy consumption after making the scheduling decision. The slack time of task $v_i$ is defined by

$$\text{Slack}(v_i) = \min_{v_j\in \text{succ}(v_i)} \begin{cases} \text{Sch}(v_j, sT) - \text{Sch}(v_i, fT) & \text{if } v_i, v_j \text{ on same processor} \\ \text{Sch}(v_j, sT) - d_{i,j} - \text{Sch}(v_i, fT) & \text{otherwise,} \end{cases} \tag{25}$$

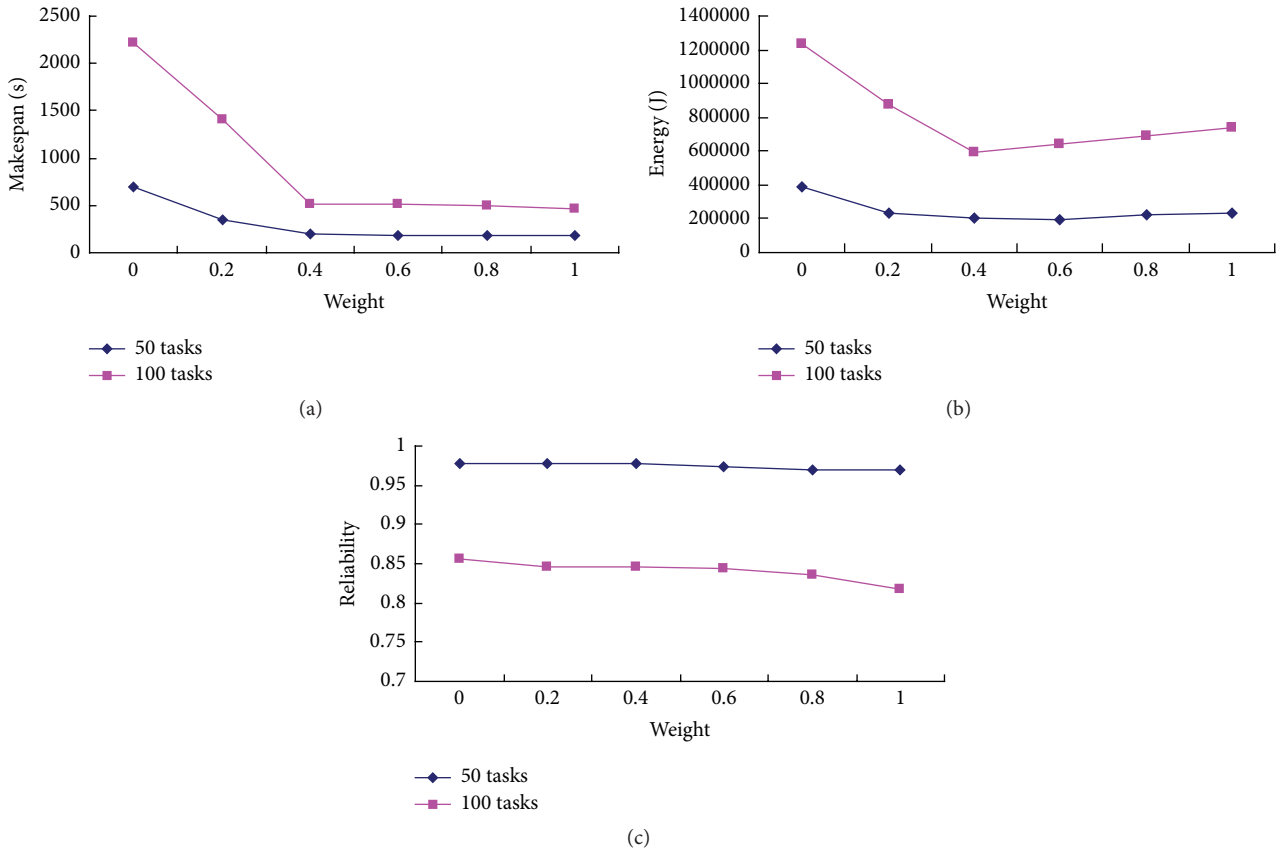


FIGURE 2: The experimental results of REAS algorithm with various weights $\theta$. (a) Schedule length. (b) Energy consumption. (c) Application reliability.

where $\mathrm{Sch}(v_i, sT)$ is the task $v_i$ earliest start time in scheduling processor-frequency pairs and $\mathrm{Sch}(v_i, fT)$ is the earliest finish time.

If task slack time $\mathrm{Slack}(v_i) > 0$, we can scale down the execution frequency to save energy consumption. Thus, the optimal frequency $f_{k,l}$ is satisfied with

$$w_{i,k,l} + \mathrm{RO}(v_i, f_{k,l}) < \mathrm{Sch}(v_i, fT) + \mathrm{Slack}(v_i),$$
$$\mathrm{EN}(v_i, f_{k,l}) < \mathrm{EN}(v_i, f_{k,\mathrm{orig}}), \quad (26)$$

where $f_{k,\mathrm{orig}}$ is the original scheduling processor-frequency pairs. At last step, we reassign task $v_i$ to the optimal frequency $f_{k,l}$.

## 6. Experimental Results and Discussion

In this section, we compare the performance, energy consumption, and system reliability using our REAS algorithm with three existing scheduling algorithms: DLS [6], RDLS [27], and ECS [20]. The experiments are performed on the synthetic randomly generated precedence-constrained parallel application graphs as described below. The performance metrics chosen for the comparison are the schedule length (2) and (22), systems energy consumption $\mathrm{EN}(P)$ (7), and application reliability $P[G]$ (16).

To test the performance of these algorithms, we have developed a discrete event simulation environment of heterogeneous systems with 8 DVFS-enabled processors using C++. This simulator includes 2 Intel® Core™ Duo, 2 Intel Xeon, 2 AMD Athlon, 1 TI DSP, and 1 Tesla GPU, mostly based on Intel processor. The systems are interconnected by Infiniband, which is a switched fabric communications link primarily used in high-performance computing. For the Infiniband configuration, the switch considered is Mellanox InfiniScaleTM III SDR and NIC is Mellanox ConnectXTM IB Dual Copper Card [21]. Other parameters of the model are set as follows. The failure rates of processors are assumed to be uniformly distributed between $1\times10^{-4}$ and $1\times10^{-5}$ failures/hr [8, 9, 28]; the transmission rates of links are assumed to be 1000 Mbits/sec.

*6.1. Randomly Generated Application Graphs.* These experiments use three commonly DAG characteristics to generate parallel application graphs [5, 8, 9, 29]:

(i) *DAG Size (v)*. It is the number of tasks in the application DAG.

(ii) *Communication-Computation Ratio (CCR)*. It is the ratio of communication time to computation time. A small CCR value means the application is computation-intensive; a large CCR value indicates

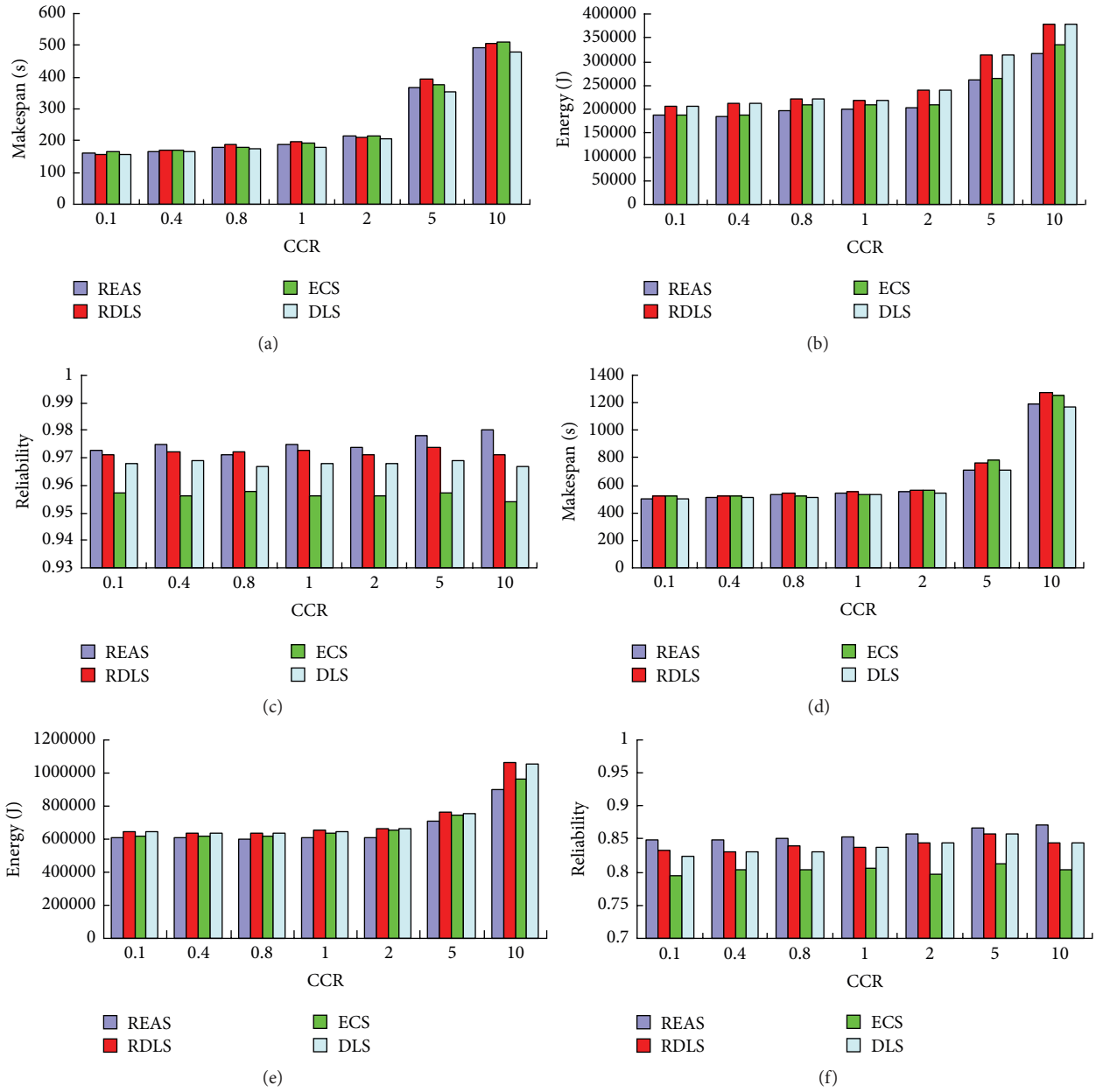


FIGURE 3: The experimental results. (a) 50-task schedule length. (b) 50-task energy consumption. (c) 50-task application reliability. (d) 100-task schedule length. (e) 100-task energy consumption. (f) 100-task application reliability.

that the application is communication-intensive [5, 8–10, 29].

(iii) *Out-Degree*. It is out-degree of a task node.

In experiments setting, DAG are generated based on the above parameters with the number of tasks 50 and 100. Task weights are generated randomly from uniform distribution $[1 \times 10^9, 9 \times 10^{11}]$ execution cycles to be around $4.5 \times 10^{10}$; thus the average task execution cycles are $4.5 \times 10^{10}$. We also generated edge weights with a uniform distribution based on a mean CCR. Different objective parallel applications can be produced as giving various CCR values [5, 8–10, 29]. In these experiments, we varied CCR in a reasonable range of 0.1 to 10.

*6.2. Various Weight $\theta$ of REAS Algorithm.* In the first experiments, we evaluate the performance of weight $\theta$ to REAS algorithm. Figure 2 shows the simulation results of scheduling 50 and 100 tasks with CCR = 1 by varying weight $\theta$ from 0 to 1, in steps of 0.2. We observe from Figure 2 that the schedule length and energy consumption decrease and the application reliability almost at the same level as the REAS algorithm weight $\theta$ increases. It is reasonable that the REAS algorithm with high $\theta$ is mostly based on task execution time and makes its schedule length shorter and consumes less

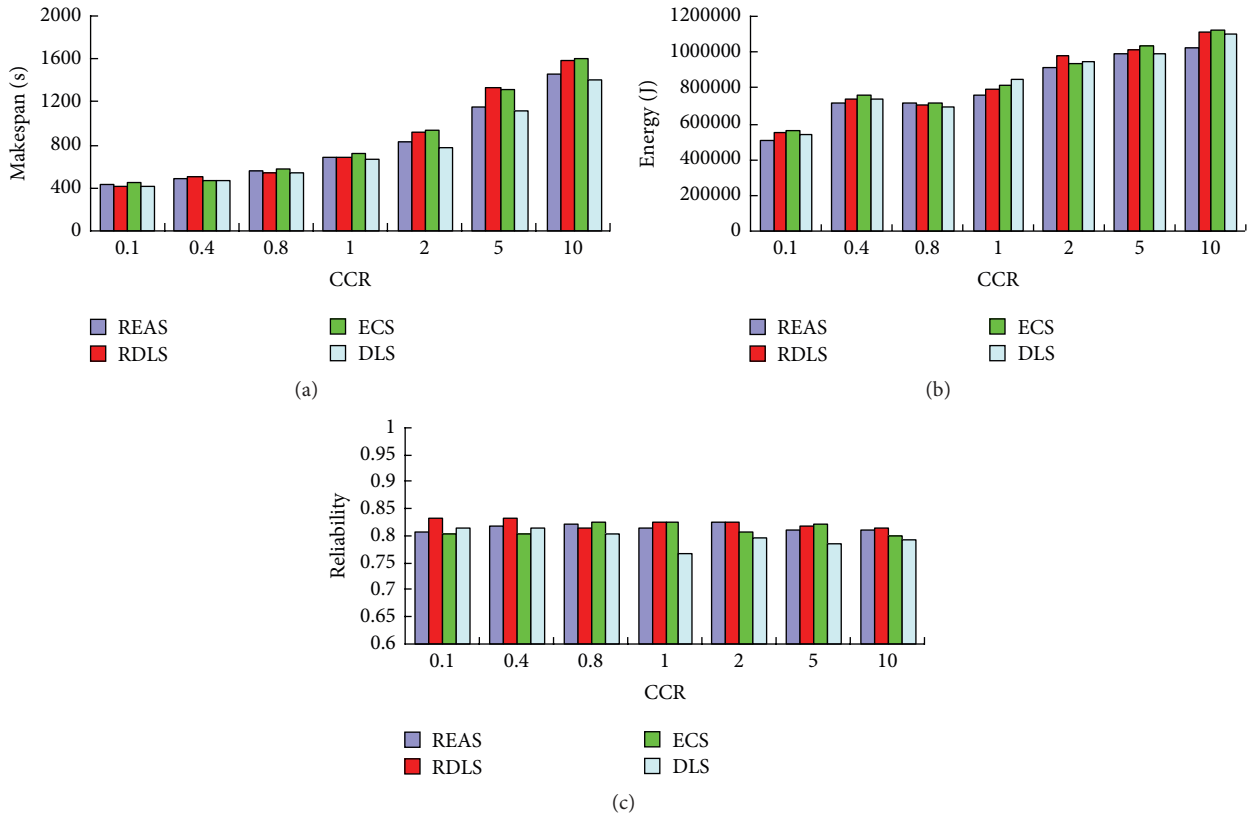


Figure 4: The experimental results of 100 tasks for 4 Intel Xeon and 4 AMD Athlon. (a) Schedule length. (b) Energy consumption. (c) Application reliability.

energy. However, as the weight $\theta$ over 0.4, the performance of REAS is not much distinguishable. Thus, in the below experiments, we let $\theta = 0.5$.

*6.3. Random Task Performance Results.* For the set of randomly generated parallel applications, the results are shown in Figures 3 and 4, where each data point is the average of the data obtained in 1,000 experiments. In this set of experiments, we assume the weight $\theta = 0.5$ of metric RE (see (24)) in REAS algorithm. In other word, the REAS algorithm has the same weight on task execution time and energy consumption. In the next section, we will examine the performance by various weights $\theta$.

We observe from Figure 3(a) that REAS is over RDLS and ECS with respect to schedule length, and the schedule length increases as the CCR increases. The average schedule length of the REAS algorithm is shorter than that of the RDLS and ECS by 2.6% and 1.9%, respectively. This improvement becomes more obvious as CCR increases, for CCR = 5 and REAS over RDLS and ECS by 7.5% and 2.6%, respectively. However, the REAS is inferior to DLS in terms of schedule length. Figure 3(b) reveals that REAS saves more average energy consumption than RDLS by 15.3%, ECS by 3.7%, and DLS by 16%, respectively. Figure 3(c) shows that REAS outperforms RDLS, ECS, and DLS by 0.3%, 2%, and 0.7% in terms of the average application reliability.

This is mainly due to the fact that REAS algorithm schedules tasks according to the novel objective RE, which can get effective tradeoff among task execution time, energy consumption, and task execution reliability. However, DLS algorithm only focuses on optimizing the task execution time and its actual execution time including the task scheduling time and reliability overhead. Thus, the scheduling solution generated by DLS can get optimal schedule length. However, it consumes more energy and has lower reliability. RDLS algorithm schedules tasks considering their execution reliability and ignoring task energy consumption. ECS algorithm is a solution for optimizing both schedule length and energy consumption, but this solution needs more task execution reliability overhead. Thus, REAS algorithm outperforms RDLS, ECS, and DLS in terms of the schedule length, energy consumption, and reliability. Other interesting experimental phenomena are that RDLS and DLS are better than ECS in terms of reliability. This is mainly due to the fact that tasks of solutions RDLS and DLS are always executing on the normal frequency of processor, which has the high reliability in all processor frequency.

The improvements of scheduling performance also could be concluded from Figures 3(d), 3(e), and 3(f) for 100 tasks. These results also show REAS over RDLS and ECS by 4.9% and 3.5% in terms of the average schedule length. And, REAS is also over RDLS, ECS, and DLS by 8.93%, 4.53%, and 8.24% in terms of the average energy consumption and by 1.86%,

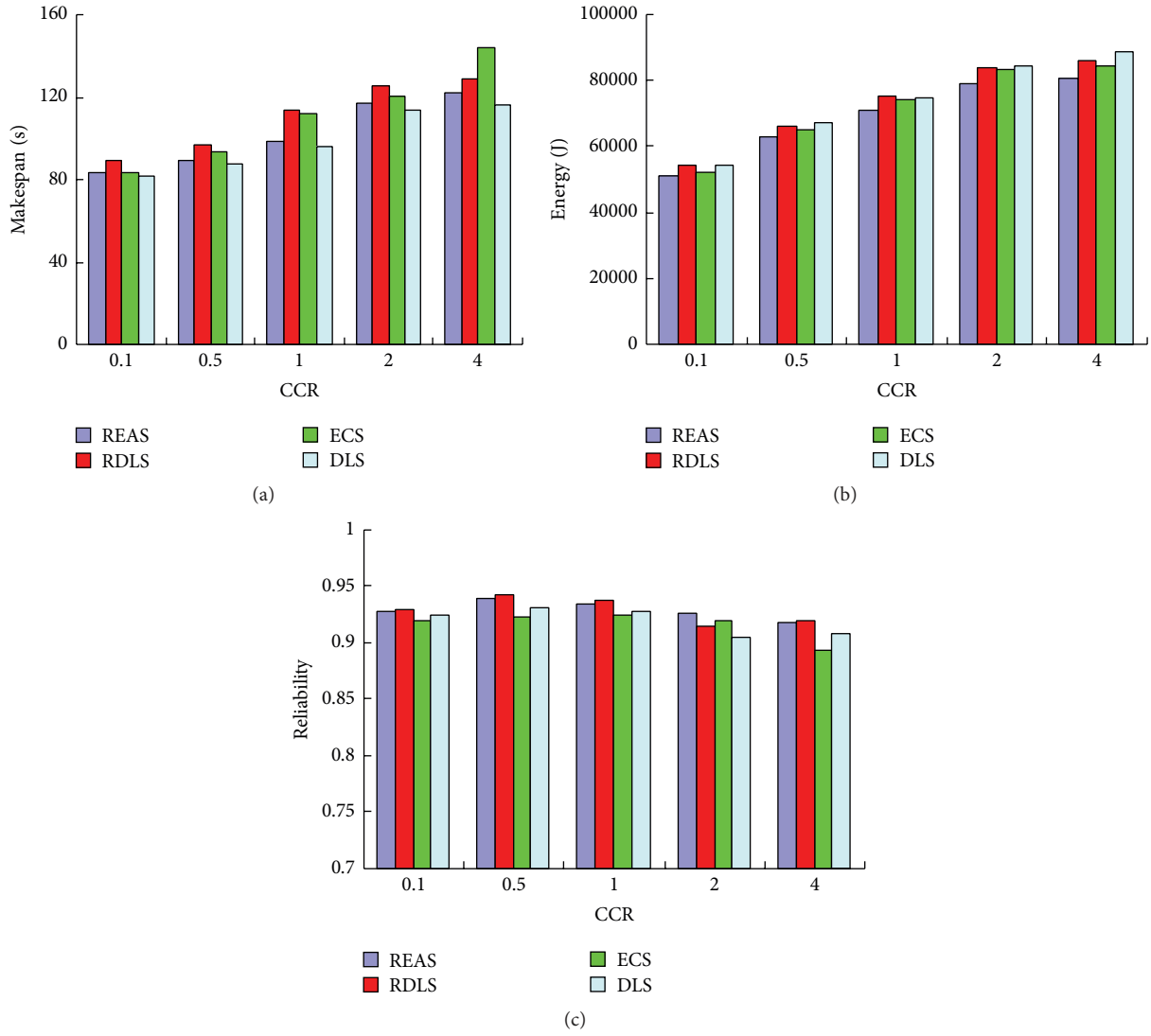


Figure 5: The experimental results of real-world DSP problem. (a) Schedule length. (b) Energy consumption. (c) Application reliability.

6.28%, and 2.1% in terms of the average application reliability, respectively.

We also simulate heterogeneous systems with 4 Intel Xeon and 4 AMD Athlon; the other configurations are the same as before. Figure 4 shows the results of 100 randomly generated tasks on this heterogeneous computing platform. The results show REAS over RDLS, ECS, and DLS in terms of average schedule length and energy consumption. However, REAS is inferior to RDLS in terms of the application reliability.

*6.4. Application Graphs of Real-World Problem.* Using real applications to test the performance of algorithms is very common [5, 8–10, 29]. In this section, we also simulate a real-world digital signal processing (DSP) problem, and the detail can be seen in [5, 8–10, 29]. From Figure 5, we can conclude that REAS is also better than RDLS, ECS, and DLS.

## 7. Conclusions and Future Work

In the past few years, with the rapid development of heterogeneous systems, the high price of energy, system performance, reliability, and various environmental issues have forced the high-performance computing sector to reconsider some of its old practices with an aim to create more sustainable system. In this paper, we attempt the simultaneous management of system performance, reliability, and energy consumption. To achieve this goal, we first built a reliability and energy aware task scheduling architecture, which mainly includes heterogeneous systems, parallel application DAG model, and energy consumption model. Then, we proposed a relationship between execution reliability and processor's voltage/frequency and deduced its parameters approximation value by least squares curve fitting method. Thirdly, we established parallel application execution reliability model and formulated this reliability and energy aware scheduling problem as a linear programming. Finally, to provide an

optimum solution for this problem, we proposed a heuristic Reliability-Energy Aware Scheduling (REAS) algorithm based on a novel scheduling objective RE, which is synthetic considering the task execution time, energy consumption, and reliability.

The performance of REAS algorithm is evaluated with an extensive set of simulations and compared to three of the best existing scheduling algorithms for heterogeneous systems: the RDLS, ECS, and DLS algorithms. The comparison is also performed on the synthetic randomly generated precedence-constrained parallel application DAG. The simulation experiment results clearly confirm the superior performance of REAS algorithm over the other three, particularly in energy saving.

This work is one of the first attempts to consider the simultaneous management of system performance, reliability, and energy consumption on high-performance computing systems. Future studies in this domain are twofold. Firstly, it will be interesting to extend our model to multidimensional computing resources, such as interconnections, memory access, and I/O activities. Secondly, in this paper, the failures occurring on resources of systems are assumed to follow Poisson process. Other reliability models can also be used in further studies.

## Competing Interests

The authors declare that they have no competing interests.

## Acknowledgments

This research was partially funded by the National Science Foundation of China (Grant no. 61370098), Hunan Provincial Natural Science Foundation of China (Grant no. 2015JJ2078), National High-Tech R&D Program of China (2015AA015303), Key Technology Research and Development Programs of Guangdong Province (2015B010108006), and a project supported by the Science Foundation for Postdoctorate Research from the Ministry of Science and Technology of China (Grant no. 2014M552134).

## References

[1] http://www.datacenterdynamics.com/focus/archive/2014/01/dcd-industry-census-2013-data-center-power.

[2] http://www.top500.org/lists/2015/11/.

[3] S. Rusu, S. Tam, H. Muljono et al., "A 45 nm 8-core enterprise xeonr processor," *IEEE Journal of Solid-State Circurits*, vol. 45, no. 1, pp. 7–14, 2010.

[4] L. Charng-Da, *Scalable diskless checkpointing for large parallel systems [Ph.D. thesis]*, University of Illinois at Urbana-Champaign, 2005.

[5] X. Tang, K. Li, Z. Zeng, and B. Veeravalli, "A novel security-driven scheduling algorithm for precedence-constrained tasks in heterogeneous distributed systems," *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 1017–1029, 2011.

[6] F. Dong and G. Selim, "Scheduling algorithms for grid computing: state of the art and open problems," Tech. Rep. 2006-504, 2006.

[7] Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Information Sciences*, vol. 270, pp. 255–287, 2014.

[8] X. Tang, K. Li, R. Li, and B. Veeravalli, "Reliability-aware scheduling strategy for heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 70, no. 9, pp. 941–952, 2010.

[9] X. Tang, K. Li, and G. Liao, "An effective reliability-driven technique of allocating tasks on heterogeneous cluster systems," *Cluster Computing*, vol. 17, no. 4, pp. 1413–1425, 2014.

[10] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[11] Y. Xu, K. Li, L. He, and T. K. Truong, "A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization," *Journal of Parallel and Distributed Computing*, vol. 73, no. 9, pp. 1306–1322, 2013.

[12] Y. Wang, K. Li, H. Chen, L. He, and K. Li, "Energy-aware data allocation and task scheduling on heterogeneous multiprocessor systems with time constraints," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 134–148, 2014.

[13] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys*, vol. 37, no. 3, pp. 195–237, 2005.

[14] K. Li, X. Tang, and Q. Yin, "Energy-aware scheduling algorithm for task execution cycles with normal distribution on heterogeneous computing systems," in *Proceedings of the 41st International Conference on Parallel Processing (ICPP '12)*, pp. 40–47, Pittsburgh, Pa, USA, September 2012.

[15] Z. Du, H. Sun, Y. He, Y. He, D. A. Bader, and H. Zhang, "Energy-efficient scheduling for best-effort interactive services to achieve high response quality," in *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS '13)*, pp. 637–648, Boston, Mass, USA, May 2013.

[16] J.-J. Han, M. Lin, D. Zhu, and L. T. Yang, "Contention-aware energy management scheme for NoC-based multicore real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 691–701, 2015.

[17] J. Mei, K. Li, and K. Li, "Energy-aware task scheduling in heterogeneous computing environments," *Cluster Computing*, vol. 17, no. 2, pp. 537–550, 2014.

[18] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz, "Bounding energy consumption in large-scale MPI programs," in *Proceedings of the ACM/IEEE Conference on Supercomputing (SC '07)*, pp. 1–9, Reno, Nev, USA, November 2007.

[19] N. B. Rizvandi, J. Taheri, and A. Y. Zomaya, "Some observations on optimal frequency selection in DVFS-based energy consumption minimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1154–1164, 2011.

[20] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1374–1381, 2011.

[21] Z. Zong, A. Manzanares, X. Ruan, and X. Qin, "EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters," *IEEE Transactions on Computers*, vol. 60, no. 3, pp. 360–374, 2011.

[22] E. Chielle, F. Lima Kastensmidt, and S. Cuenca-Asensi, "Tuning software-based fault-tolerance techniques for power optimization," in *Proceedings of the 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS '14)*, pp. 1–7, Palma de Mallorca, Spain, October 2014.

[23] D. Ernst, S. Das, S. Lee et al., "Razor: circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, 2004.

[24] F. Firouzi, A. Yazdanbakhsh, H. Dorosti, and S. M. Fakhraie, "Dynamic soft error hardening via joint body biasing and dynamic voltage scaling," in *Proceedings of the 14th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD '11)*, pp. 385–392, Oulu, Finland, March 2011.

[25] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 35–40, IEEE, November 2004.

[26] D. Zhu and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 58, no. 10, pp. 1382–1397, 2009.

[27] A. Dogan and F. Özgüner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 308–323, 2002.

[28] X. Qin and H. Jiang, "A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters," *Journal of Parallel and Distributed Computing*, vol. 65, no. 8, pp. 885–900, 2005.

[29] Y. Xu, K. Li, L. He, L. Zhang, and K. Li, "A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems," *IEEE Transaction on Parallel and Distributed System*, vol. 26, no. 12, pp. 3208–3222, 2015.

[30] V. Spiliopoulos, S. Kaxiras, and G. Keramidas, "Green governors: a framework for continuously adaptive DVFS," in *Proceedings of the International Green Computing Conference (IGCC '11)*, pp. 1–8, IEEE, Orlando, Fla, USA, July 2011.

[31] M. Qiu and E. H.-M. Sha, "Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, article 25, 2009.

[32] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers," *Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 732–749, 2011.

[33] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *Proceedings of the International Electron Devices Meeting (IEDM '02)*, pp. 329–332, San Francisco, Calif, USA, December 2002.

[34] A. M. Fard, M. Ghasemi, and M. Kargahi, "Response-time minimization in soft real-time systems with temperature-affected reliability constraint," in *Proceedings of the CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST '15)*, Tehran, Iran, October 2015.

[35] S. Song, D. W. Coit, Q. Feng, and H. Peng, "Reliability analysis for multi-component systems subject to multiple dependent competing failure processes," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 331–345, 2014.

[36] V. Degalahal, L. Li, V. Narayanan, M. Kandemir, and M. J. Irwin, "Soft errors issues in low-power caches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1157–1166, 2005.

[37] Z. Lei, G. Tianqi, Z. Ji, J. Shijun, S. Qingzhou, and H. Ming, "An adaptive moving total least squares method for curve fitting," *Measurement*, vol. 49, pp. 107–112, 2014.