Received: 2 May 2016 | Revised: 17 August 2016 | Accepted: 23 September 2016
DOI 10.1002/cpe.4024

WILEY

SPECIAL ISSUE PAPER

Schedule length minimization of parallel applications with energy consumption constraints using heuristics on heterogeneous distributed systems

Guoqi Xie[1,2] | Xiongren Xiao[1,2] | Renfa Li[1,2] | Keqin Li[1,3]

[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha, China
[2]Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha, China
[3]Department of Computer Science, State University of New York, Albany, NY, USA

Correspondence
Guoqi Xie, College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China.
Email: xgqman@hnu.edu.cn

Funding Information
National Key Research and Development Plan of China, Grant/Award Number: 2012AA01A301-01 and 2016YFB0200405; National Natural Science Foundation of China, Grant/Award Number: 61432005, 61173036, 61300037, 61370095, 61300039, 61502162, 61379115, 61502405, 61402170 and 61370097; China Postdoctoral Science Foundation, Grant/Award Number: 2016M592422

Summary

Energy consumption is one of the primary design constraints in heterogeneous parallel and distributed systems ranging from small embedded devices to large-scale data centers. The problem of minimizing the schedule length of an energy consumption-constrained parallel application has been studied recently in homogeneous systems with a shared memory. To adopt the heterogeneity and distribution of high-performance computing systems, this study solves the problem of minimizing the schedule length of an energy consumption-constrained parallel application in heterogeneous distributed systems based on a dynamic voltage and frequency scaling energy-efficient design technique. The aforementioned problem is divided into 2 subproblems in this study, namely, satisfying energy consumption constraint and minimizing schedule length. The first subproblem is solved by transferring the energy consumption constraint of the application to that of each task, whereas the second subproblem is solved by heuristically scheduling each task with low time complexity. Experiments using both fast Fourier transform and Gaussian elimination parallel applications show that the actual energy consumption values do not always exceed but are close to the given energy consumption constraints. In addition, the minimum schedule lengths are generated using the proposed algorithm.

KEYWORDS
energy consumption, heterogeneous systems, parallel applications, schedule length

1 | INTRODUCTION

1.1 | Background

Recent trends in the microprocessor industry have important implications for the design of high-performance computing systems. Performance is improved while keeping energy consumption to a minimum by increasing the number of heterogeneous processors and cores. This trend has reached the deployment stage in heterogeneous parallel and distributed systems, which range from small embedded devices to large-scale data centers. A number of heterogeneous processors and cores in these systems are expected to increase dramatically in the near future. For these systems, energy consumption is one of the primary design constraints. The popular energy consumption optimization technique, namely, dynamic voltage and frequency scaling (DVFS), achieves energy-efficient optimization by simultaneously scaling down the supply voltage and frequency of a processor while tasks are running to explore the trade-off between energy consumption and execution time.[1–3]

A parallel application with precedence-constrained tasks at a high level of heterogeneous distributed systems is described by a directed acyclic graph (DAG),[4–6] where nodes represent tasks and edges represent communication messages between tasks. Reducing the scheduling length (also called makespan) for fastest execution of a DAG-based parallel application is the main concern in system performance.[4,7–9] The schedule length is represented as the time span that from the start time instant of first task to the finish time instant of the last task. Scheduling tasks on heterogeneous processors with the objective of minimizing the schedule length of a DAG-based parallel application is a well-known nondeterministic polynomial–hard optimization problem, and numerous heuristic list scheduling algorithms have been proposed to generate near-optimal solutions of the problem.[4–6] Similarly, minimizing schedule length of a DAG-based parallel application with energy consumption constraint on heterogeneous distributed systems is also a nondeterministic polynomial–hard optimization problem; however, to the best of our knowledge, this problem has not been studied before.

1.2 | Related work

An energy-efficient design technique based on DVFS was first introduced in the work of Weiser et al.[10] This work inspired substantial further investigations on energy consumption optimization for scheduling independent or precedence-constrained tasks on a uniprocessor or multiprocessors.[2,3,11–15] Li[16] studied the energy-aware task scheduling of independent sequential tasks in homogeneous multiprocessors as combinatorial optimization problems. Rusu et al[17] simultaneously addressed 3 constraints (ie, energy, deadline, and reward) in both homogeneous and heterogeneous systems. Li et al[18] studied the problem of scheduling a collection of independent tasks with deadlines and energy consumption constraints in heterogeneous systems.

The aforementioned studies are restricted to independent tasks. However, parallel applications, such as fast Fourier transform (FFT) and Gaussian elimination applications[2,4] with precedence-constrained tasks, are widely used in high-performance heterogeneous distributed computing systems. Zong et al[19] considered energy-aware duplication scheduling algorithms for a parallel application in homogeneous systems. Lee and Zomaya[20] presented the energy-conscious scheduling (ECS) algorithm to implement joint minimization between schedule length and energy consumption for a parallel application in heterogeneous distributed systems. Khan and Zomaya[2] and Li[3] addressed the problem of minimizing energy consumption with a schedule length constraint for a parallel application in homogeneous systems. The problem of minimizing the schedule length of an energy consumption-constrained application with precedence-constrained sequential[1] and parallel tasks (ie, a parallel application)[2,21] was solved. These 2 studies were interested only in homogeneous systems with a shared memory and could not be applied to heterogeneous distributed systems with communication time between any 2 tasks. The current study aims to achieve the objective of minimizing the schedule length of an energy consumption-constrained parallel application in heterogeneous distributed systems.

Other related studies also involve the schedule length or energy consumption. Arabnejad and Barbosa[22] presented a budget constrained scheduling algorithm for parallel applications on heterogeneous systems. Zhang et al[23] studied the problem of maximizing the reliability of energy consumption-constrained parallel applications on heterogeneous systems. Huang et al[24] and Tang et al[25] studied the problem of minimizing energy consumption of a schedule length constrained parallel application on heterogeneous systems. Zhang et al[26] studied the deadline-driven flow scheduling scheme in multi-resource environments.

1.3 | Our contributions

The problem of minimizing the schedule length of an energy consumption-constrained parallel application in heterogeneous distributed systems is divided into 2 subproblems, namely, satisfying energy consumption constraint and minimizing schedule length. The contributions of this study are summarized as follows.

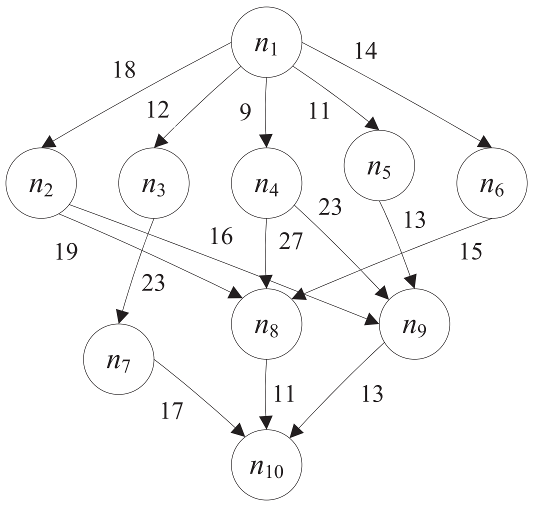
$$\begin{tikzcd}
 & & n_1 \arrow[dll, "18"'] \arrow[dl, "12"'] \arrow[d, "9"] \arrow[dr, "11"] \arrow[drr, "14"] & & \\
n_2 \arrow[drr, "19"'] \arrow[drrr, "16"] & n_3 \arrow[d, "23"] & n_4 \arrow[d, "27"] \arrow[dr, "23"] & n_5 \arrow[d, "13"] & n_6 \arrow[dll, "15"] \\
 & n_7 \arrow[dr, "17"'] & n_8 \arrow[d, "11"] & n_9 \arrow[dl, "13"] & \\
 & & n_{10} & &
\end{tikzcd}$$

FIGURE 1 A motivating example of a directed acyclic graph–based parallel application with 10 tasks[4–6]

1. We solve the first subproblem. That is, the energy consumption constraint of the application can always be satisfied at each task assignment phase by presupposing that the unassigned tasks are assigned to the processor with the minimum energy consumption. Consequently, the energy consumption constraint of the application is transferred to that of each task.
2. We solve the second subproblem. That is, the schedule length of the application can be minimized by heuristically scheduling each task with low time complexity. Hence, the earliest finish time (EFT) is assigned to the processor.
3. We perform extensive experiments with FFT and Gaussian elimination parallel applications. We determine that actual energy consumption values do not always exceed but are close to the given energy consumption constraints. Moreover, shorter schedule lengths are generated using the proposed algorithm.

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

2 | MODELS

2.1 | Application model

This study considers a system platform with fully connected heterogeneous multiprocessors. Let $U = \{u_1, u_2, \dots, u_{|U|}\}$ represents a set of heterogeneous processors, where $|U|$ denotes the size of set U. For any set X, this study uses $|X|$ to denote its size. A parallel application running on processors is represented using a DAG $G = (N, M, C, W)$.[4–7,20] N represents a set of nodes in G, and each node $n_i \in N$ indicates a task with various execution time on different processors. M is a set of communication edges, and each edge $m_{i,j} \in M$ refers to the communication message from n_i to n_j. Accordingly, $c_{i,j} \in C$ represents the communication time of $m_{i,j}$ if n_i and n_j are not assigned to the same processor. $pred(n_i)$

TABLE 1 Execution time of tasks on different processors with the maximum frequency of the parallel application in Figure 1[4–6]

Task	u_1	u_2	u_3	$rank_u$
n_1	14	16	9	108.000
n_2	13	19	18	77.000
n_3	11	13	19	80.000
n_4	13	8	17	80.000
n_5	12	13	10	69.000
n_6	13	16	9	63.333
n_7	7	15	11	42.667
n_8	5	11	14	35.667
n_9	18	12	20	44.333
n_{10}	21	7	16	14.667

represents the set of immediate predecessor tasks of n_i. $succ(n_i)$ represents the set of immediate successor tasks of n_i. The task without a predecessor is denoted as n_{entry}, whereas the task without a successor is denoted as n_{exit}. W is a $|N| \times |U|$ matrix, where $w_{i,k}$ indicates the execution time of n_i running on u_k with the maximum frequency.[7]

Figure 1 shows a standard motivating example of a DAG-based parallel application. This example has been used in numerous studies.[4–6] Table 1 is a matrix of execution time with the maximum frequency in Figure 1. The example shows 10 tasks executed on 3 processors $\{u_1, u_2, u_3\}$. Weight 14 of n_1 and u_1 in Table 1 represents the execution time denoted by $w_{1,1} = 14$. Notably, the same task has varying execution time on different processors because of the heterogeneity of the processors.[4] Weight 18 of edge (Figure 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.[6]

2.2 | Power and energy model

Given the nearly linear relationship between voltage and frequency, DVFS scales down voltage alongside frequency to save energy. Similar to the works of Zhu and Aydin[27] and Zhao et al,[28] we use the term frequency change to refer to simultaneously changing voltage and frequency. Considering a DVFS-capable system, we also adopt the system-level power model that is widely used in 2 studies,[27,28] where power consumption at frequency f is given by

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

P_s represents the static power and can only be removed by powering off the whole system. P_{ind} refers to frequency-independent dynamic power, which can be removed by putting the system in sleep mode. P_d denotes frequency-dependent dynamic power that depends on frequencies. h represents system state; it indicates whether dynamic power is currently consumed in the system. When the system is active, $h = 1$; otherwise, $h = 0$. C_{ef} represents effective switching capacitance, and m denotes the dynamic power exponent that is greater than 2. Both C_{ef} and m are processor-dependent constants.

Notably, an excessive overhead that is associated with turning a system on/off exists. P_s is always consumed and unmanageable.[27,28] Similar to the aforementioned studies, the current study focuses on managing dynamic power (ie, P_{ind} and P_d). Given that P_{ind}, less P_d does not result in less energy consumption. That is, a minimum energy-efficient frequency f_{ee} exists[27,28] and is denoted as

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

Assuming that the frequency of a processor varies from a minimum available frequency f_{min} to the maximum frequency f_{max}, the lowest frequency that should execute a task should be $f_{low} = \max(f_{min}, f_{ee})$. Hence, any actual effective frequency f_h should belong to the scope of $f_{low} \leqslant f_h \leqslant f_{max}$.

Given that the number of processors in the system is $|U|$, and these processors are completely heterogeneous, each processor should have individual power parameters. In this study, we define a frequency-independent dynamic power set as

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

a frequency-dependent dynamic power set as

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

an effective switching capacitance set as

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

a dynamic power exponent set as

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

a minimum energy-efficient frequency set as

$$\{f_{1,ee}, f_{2,ee}, \ldots, f_{|U|,ee}\},$$

and an actual effective frequency set as

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

Then, let $E(n_i, u_k, f_{k,h})$ represents the processor energy consumption of task n_i on processor u_k with frequency $f_{k,h}$, which is calculated as

$$E(n_i, u_k, f_{k,h}) = P_{k,h} \times w_{i,k} \times \frac{f_{k,max}}{f_{k,h}}, \tag{2}$$

where

$$P_{k,h} = \left(P_{k,ind} + C_{k,ef} \times (f_{k,h})^{m_k}\right) \tag{3}$$

represents the dynamic power of processor u_k with frequency $f_{k,h}$.

3 | PRELIMINARIES

3.1 | Energy consumption constraint

The execution time of each task on each processor is known; hence, we can determine the minimum and maximum energy consumption denoted by $E_{min}(n_i)$ and $E_{max}(n_i)$, respectively, by traversing all the processors. $E_{min}(n_i)$ and $E_{max}(n_i)$ are obtained by executing the tasks with the maximum and minimum frequencies, respectively. These variables are calculated using

$$E_{min}(n_i) = \min_{u_k \in U} E(n_i, u_k, f_{k,max}), \tag{4}$$

and

$$E_{\max}(n_i) = \max_{u_k \in U} E(n_i, u_k, f_{k,\mathrm{ee}}), \tag{5}$$

respectively.

The energy consumption of application G is the sum of the energy consumption of each task; thus, we can obtain the minimum and maximum energy consumption of G as

$$E_{\min}(G) = \sum_{i=1}^{|N|} E_{\min}(n_i), \tag{6}$$

and

$$E_{\max}(G) = \sum_{i=1}^{|N|} E_{\max}(n_i), \tag{7}$$

respectively.

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

$$E_{\min}(G) \leqslant E_{\mathrm{given}}(G) \leqslant E_{\max}(G). \tag{8}$$

3.2 | Problem description

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

$$SL(G) = AFT(n_{\mathrm{exit}}),$$

where $AFT(n_{\mathrm{exit}})$ represents the actual finish time (AFT) of the exit task n_{exit} subject to its energy consumption constraint as follows:

$$E(G) = \sum_{i=1}^{|N|} E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leqslant E_{\mathrm{given}}(G), \tag{9}$$

where $u_{pr(i)}$ and $f_{pr(i),hz(i)}$ represent the assigned processor and frequency of n_i, respectively, and $f_{pr(i),\mathrm{low}} \leqslant f_{pr(i),hz(i)} \leqslant f_{pr(i),\max}$, for $\forall i{:}1 \leqslant i \leqslant |N|, u_{pr(i)} \in U$.

3.3 | Task prioritization

The task assignment order should be determined first before assigning tasks to processors. Similar to the works of Topcuoglu et al[4] and Lee and Zomaya,[20] the upward rank value ($rank_{\mathrm{u}}$) of a task given by Equation (10) is used as the common task priority standard. All the tasks are arranged according to the decreasing order of $rank_{\mathrm{u}}$.

$$rank_{\mathrm{u}}(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{c_{i,j} + rank_{\mathrm{u}}(n_j)\}, \tag{10}$$

where $\overline{w_i}$ represents the average execution time of task n_i, which is calculated as $\overline{w_i} = \left(\sum_{k=1}^{|U|} w_{i,k}\right) / |U|$. Table 1 also shows the upward rank values of all the tasks (Figure 1). Notably, n_i is prepared to be assigned only if all the predecessors of n_i have been assigned to the processors. Assume that 2 tasks n_i and n_j satisfy $rank_{\mathrm{u}}(n_i) > rank_{\mathrm{u}}(n_j)$. If no precedence constraint exists between n_i and n_j, then n_i does not necessarily take precedence n_j to be assigned. We can draw the conclusion that the task assignment order in G is $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$.

4 | SCHEDULING POLICY

The problem of minimizing the schedule length of an energy consumption-constrained parallel application in heterogeneous distributed systems is divided into 2 subproblems, namely, satisfying energy consumption constraint and minimizing schedule length. We first solve these 2 subproblems individually and then present the algorithm by integrating the 2 subproblems.

4.1 | Satisfying the energy consumption constraint

Assume that the task to be assigned is $n_{seq(j)}$, where $seq(j)$ represents the j-th assigned task (sequence number). Then, $\{n_{seq(1)}, n_{seq(2)}, \ldots, n_{seq(j-1)}\}$ represents the task set where the tasks have been assigned, whereas $\{n_{seq(j+1)}, n_{seq(j+2)}, \ldots, n_{seq(|N|)}\}$ represents the task set where the tasks have not been assigned. To ensure that the energy consumption constraint of the application is satisfied at each task assignment, we presuppose that each task in $\{n_{seq(j+1)}, n_{seq(j+2)}, \ldots, n_{seq(|N|)}\}$ is assigned to the processor and frequency with the minimum energy consumption. Hence, when assigning $n_{seq(j)}$, the energy consumption of G is calculated as

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

For any task $n_{seq(j)}$, the actual energy consumption $E(G) = \sum_{i=1}^{|N|} E(n_i, u_{pr(i)}, f_{pr(i),hz(i)})$ should be less than or equal to $E_{\mathrm{given}}(G)$ only if $E_{seq(j)}(G) \leqslant E_{\mathrm{given}}(G)$. We prove the validity of this strategy in the following paragraphs.

Theorem 1. Each task $n_{seq(j)}$ in parallel application G can always find an assigned processor and a corresponding frequency to satisfy

$$\begin{aligned} E_{seq(j)}(G) = &\sum_{x=1}^{j-1} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \\ &+ E(n_{seq(j)}, u_k, f_{k,h}) + \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}) \leqslant E_{\mathrm{given}}(G). \end{aligned} \tag{11}$$

Proof. We apply mathematical induction for the proof and first consider the entry task $n_1 = n_{seq(1)}$. In this case, all the tasks are not assigned to processors, and application G should satisfy its energy consumption constraint:

$$E_{seq(1)}(G) = E(n_{seq(1)}, u_k, f_{k,h}) + \sum_{y=2}^{|N|} E_{\min}(n_{seq(y)}) \leqslant E_{\mathrm{given}}(G), \tag{12}$$

that is, $n_{seq(1)}$ is required to satisfy

$$E(n_{seq(1)}, u_k, f_{k,h}) \leqslant E_{\mathrm{given}}(G) - \sum_{y=2}^{|N|} E_{\min}(n_{seq(y)}). \tag{13}$$

Given that

$$E_{\min}(G) = E_{\min}(n_{seq(1)}) + \sum_{y=2}^{|N|} E_{\min}(n_{seq(y)}) \leqslant E_{\text{given}}(G),$$

then according to Equations (6) and (8), we obtain

$$E_{\min}(n_{seq(1)}) \leqslant E_{\text{given}}(G) - \sum_{y=2}^{|N|} E_{\min}(n_{seq(y)}).$$

The minimum value of $E(n_{seq(1)}, u_k, f_{k,h})$ is $E_{\min}(n_{seq(1)})$; thus, $n_{seq(1)}$ can find an assigned processor to satisfy Equation (13); that is, Equation (12) is satisfied

$$E_{seq(1)}(G) = E(n_{seq(1)}, u_k, f_{k,h}) + \sum_{y=2}^{|N|} E_{\min}(n_{seq(y)}) \leqslant E_{\text{given}}(G).$$

Assume that j-th task $n_{seq(j)}$ can find an assigned processor $u_{pr(seq(j))}$ and frequency $f_{pr(seq(j)),hz(seq(j))}$ to satisfy $E_{\text{given}}(G)$, then we have

$$\begin{aligned} E_{seq(j)}(G) = &\sum_{x=1}^{j-1} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \\ &+ E(n_{seq(j)}, u_{pr(seq(j))}, f_{pr(seq(j)),hz(seq(j))}) \\ &+ \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}) \leqslant E_{\text{given}}(G) \end{aligned}, \tag{14}$$

that is,

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

Hence, we have

$$\begin{aligned} &\sum_{x=1}^{j} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \\ &\leqslant E_{\text{given}}(G) - \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}) \end{aligned}. \tag{15}$$

For $(j+1)$-th task $n_{seq(j+1)}$, the energy consumption of the application is

$$\begin{aligned} E_{seq(j+1)}(G) = &\sum_{x=1}^{j} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \\ &+ E(n_{seq(j+1)}, u_k, f_{k,h}) + \sum_{y=j+2}^{|N|} E_{\min}(n_{seq(y)}) \end{aligned}.$$

Given that $\sum_{x=1}^{j} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \leqslant E_{\text{given}}(G) - \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)})$ (Equation (15)), then we have

$$\begin{aligned} E_{seq(j+1)}(G) &\leqslant E_{\text{given}}(G) - \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}) \\ &+ E(n_{seq(j+1)}, u_k, f_{k,h}) + \sum_{y=j+2}^{|N|} E_{\min}(n_{seq(y)}) \\ &= E_{\text{given}}(G) - E_{\min}(n_{seq(j+1)}) + E(n_{seq(j+1)}, u_k, f_{k,h}) \end{aligned}. \tag{16}$$

The minimum value of $E(n_{seq(j+1)}, u_k, f_{k,h})$ is $EC_{\min}(n_{seq(j+1)})$ when $E(n_{seq(j+1)}, u_k, f_{k,h}) = E_{\min}(n_{seq(j+1)})$; hence, we have

$$E_{seq(j+1)}(G) \leqslant E_{\text{given}}(G),$$

based on Equation (16). That is, $n_{seq(j+1)}$ can also find an assigned processor to satisfy $E_{\text{given}}(G)$. Given that all the tasks can find individual assigned processors to satisfy $E_{\text{given}}(G)$, then Theorem 1 is satisfied. □

4.2 | Minimizing schedule length

Heterogeneous EFT (HEFT) is a well-known precedence-constrained task-scheduling algorithm based on the DAG model. It is used to reduce schedule length to a minimum value, which is then combined with low complexity and high performance in heterogeneous systems.[4,7] In addition to task prioritization based on the upward rank value, task assignment based on EFT is also presented because it can satisfy the local optimal of each precedence-constrained task using the greedy policy. The original EFT does not consider frequency adjustment, and thus, a new EFT should be presented.

Let $EST(n_i, u_k, f_{k,h})$ and $EFT(n_i, u_k, f_{k,h})$ represent the earliest start time and EFT, respectively, of task n_i on processor u_k with frequency $f_{k,h}$. The aforementioned attributes are calculated as

$$\begin{cases} EST(n_{\text{entry}}, u_k, f_{k,h}) = 0 \\ EST(n_i, u_k, f_{k,h}) = \max(avail[k], \max\limits_{n_x \in pred(n_i)} \{AFT(n_x) + c'_{x,i}\} \end{cases} \tag{17}$$

and

$$EFT(n_i, u_k, f_{k,h}) = EST(n_i, u_k, f_{k,h}) + w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{18}$$

$avail[k]$ is the earliest available time when processor u_k is ready for task execution, and $AFT(n_x)$ is the AFT of n_x as mentioned earlier. $c'_{x,i}$ represents the actual communication time between n_x and n_i. If n_x and n_i are assigned to the same processor, then $c'_{x,i} = 0$; otherwise, $c'_{x,i} = c_{x,i}$. n_i is assigned to the processor with minimum EFT using the insertion-based scheduling strategy, where n_i can be inserted into the slack with the minimum EFT.

4.3 | Scheduling algorithm

We first provide the energy consumption constraint of each task before we propose the algorithm. From Equation (11), we have

$$\begin{aligned} E(n_{seq(j)}, u_k, f_{k,h}) &\leqslant E_{\text{given}}(G) \\ &- \sum_{x=1}^{j-1} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \\ &- \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}) \end{aligned}. \tag{19}$$

Hence, we let the energy consumption constraint of task $n_{seq(y)}$ be

$$\begin{aligned} E_{\text{given}}(n_{seq(j)}) = E_{\text{given}}(G) &- \sum_{x=1}^{j-1} E(n_{seq(x)}, u_{pr(seq(x))}, f_{pr(seq(x)),hz(seq(x))}) \\ &- \sum_{y=j+1}^{|N|} E_{\min}(n_{seq(y)}), \end{aligned} \tag{20}$$

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

$$E(n_{seq(j)}, u_k, f_{k,h}) \leqslant E_{\text{given}}(n_{seq(j)}). \tag{21}$$

Hence, when assigning task $n_{seq(j)}$, we can directly consider the energy consumption constraint $E_{\text{given}}(n_{seq(j)})$ of $n_{seq(j)}$ and disregard the energy consumption constraint of application G. In this manner, a low time complexity heuristic algorithm can be established. Given that the maximum energy consumption constraint of $n_{seq(j)}$ is $E_{\max}(n_i)$, $E_{\text{given}}(n_{seq(j)})$ should be required to satisfy the following constraint:

$$E(n_{seq(j)}, u_k, f_{k,h}) \leqslant \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}. \tag{22}$$

Inspired by the preceding analysis, we propose the algorithm called minimum schedule length with energy consumption constraint (MSLECC) to minimize schedule length while satisfying the energy consumption constraint of the application. The steps of MSLECC are described in Algorithm 1.

Algorithm 1 The MSLECC Algorithm

```
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:   $n_i$ = downward_task_list.out();
4:   Calculate $E_{\min}(n_i)$ and $E_{\max}(n_i)$ using Equations 4 and 5, respectively;
5:   Calculate $E_{\text{given}}(n_i)$ using Equation 19;
6:   var $pr(i) = NULL$, $f_{pr(i),hz(i)} = NULL$, $AFT(n_i) = \infty$, $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = 0$;
7:   for (each processor $u_k \in U$) do
8:     for (each frequency $f_{k,h}$ in the scope of $[f_{k,\text{low}}, f_{k,\max}]$) do
9:       Calculate $E(n_i, u_k, f_{k,h})$ using Equation 2;
10:      if ($E(n_i, u_k, f_{k,h}) > \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}$) then
11:        continue; // skip the processor and frequency that do not satisfy the energy consumption constraint of $n_i$
12:      end if
13:      Calculate $EFT(n_i, u_k, f_{k,h})$ using Equation 18;
14:      if ($EFT(n_i, u_k, f_{k,h}) < AFT(n_i)$) 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:        $AFT(n_i) = EFT(n_i, u_k, f_{k,h})$; // select the processor and frequency with the minimum EFT
19:      end if
20:    end for
21:  end for
22: end while
23: Calculate the actual energy consumption $E(G)$ using Equation 9;
24: Calculate $SL(G) = AFT(n_{\text{exit}})$;
```

The main idea of MSLECC is that the energy consumption constraint of the application is transferred to that of each task. Each task simply selects the processor and frequency with the minimum EFT to satisfy its energy consumption constraint. The core details are explained as follows.

1. In line 6, we initialize $AFT(n_i) = \infty$ and $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = 0$.
2. In lines 7Ċ21, we traverse all the processors and frequencies and then select the processor with the minimum EFT for each task to satisfy the condition of $E(n_i, u_k, f_{k,h}) \leqslant \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}$.
3. In lines 23 and 24, we calculate the actual energy consumption $E(G)$ and the final schedule length $SL(G)$, respectively.
4. Minimum schedule length with energy consumption constraint is a heuristic algorithm, and thus, it has a low time complexity of $O(|N|^2 \times |U| \times |F|)$, where F represents the maximum number of discrete frequencies from the lowest to the highest actual effective frequencies. That is, MSLECC implements low time complexity and high-performance scheduling for energy consumption-constrained parallel applications.

TABLE 2 Power parameters of processors (u_1, u_2, and u_3)

u_k	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	m_k	$f_{k,\text{ee}}(f_{k,\text{low}})$	$f_{k,\max}$
u_1	0.03	0.8	2.9	0.26	1.0
u_2	0.04	0.8	2.5	0.26	1.0
u_3	0.07	1.0	2.5	0.29	1.0

4.4 Example of the MSLECC algorithm

This section provides an example to illustrate the results using the MSLECC algorithm. We assume that the power parameters of all the processors are known and are shown in Table 2, where the maximum frequency $f_{k,\max}$ for each processor is 1 and the frequency precision is set to 0.01.

We can obtain the minimum energy-efficient frequency $f_{k,\text{ee}}$ (considered as $f_{k,\text{low}}$ in this example) for each processor and the dynamic power of $p_{k,h}$ using Equations (1) and (3), respectively.

We can calculate the minimum and maximum reliability values to be $E_{\min}(G) = 20.31$ and $E_{\max}(G) = 161.99$ using Equations (6) and (7), respectively. We set the energy consumption constraint of G to $E_{\text{given}}(G) = 0.5 \times E_{\max}(G) = 80.995$. Table 3 provides the task assignment of the parallel application in Figure 1 using MSLECC, where each row represents a task assignment and all the tasks satisfy their individual energy consumption constraints. Finally, the actual consumed energy of the application is determined as $E(G) = 80.9939$, which is less than and close to $E_{\text{given}}(G) = 80.995$. The final schedule length is $SL(G) = 129.3660$. This example also verifies that using MSLECC can ensure that the actual consumed energy does not exceed the given energy consumption constraint, namely, $E(G) \leqslant E_{\text{given}}(G)$.

Figure 2 also shows the scheduling of parallel application G in Figure 1 using MSLECC, where the schedule length is 121.84. The arrows in Figure 2 represent the generated communication time between tasks. u_3 has a considerable slack because if n_7 is assigned to u_3, then the energy consumption constraint of n_7 cannot be satisfied.

5 EXPERIMENTS

5.1 Experimental metrics

The performance metrics selected for comparison are the actual energy consumption $E(G)$ (Equation (9)) and the final schedule length $SL(G)$ of the application. The algorithms compared with our proposed MSLECC are HEFT[4] and ECS[20] because all 3 algorithms have the same application model. The processor and application parameters are as

TABLE 3 Task assignment of the application in Figure 1 using minimum schedule length with energy consumption constraint

n_i	$E_{given}(n_i)$	$u_{pr(i)}$	$f_{pr(i),hz(i)}$	$E(n_i, pr(i), f_{pr(i),hz(i)})$	$AST(n_i)$	$AFT(n_i)$
n_1	13.44	u_3	1.0	9.63	0	12
n_3	20.33	u_3	1.0	20.33	9	28
n_4	18.19	u_2	1.0	6.72	18	26
n_2	19.26	u_1	1.0	10.79	27	40
n_5	10.92	u_3	1.0	10.7	28	38
n_6	13.44	u_2	1.0	13.44	26	42
n_9	5.4385	u_2	0.61	5.3606	56	75.67
n_7	1.3188	u_1	0.33	1.3177	51	72.2121
n_8	0.8874	u_1	0.26	0.8863	72.2121	91.4429
n_{10}	1.8204	u_2	0.26	1.8193	102.4429	129.3660

$E(G) = 80.98 \leqslant E_{given}(G) = 80.9939$, $SL(G) = AFT(n_{10}) = 129.3660$

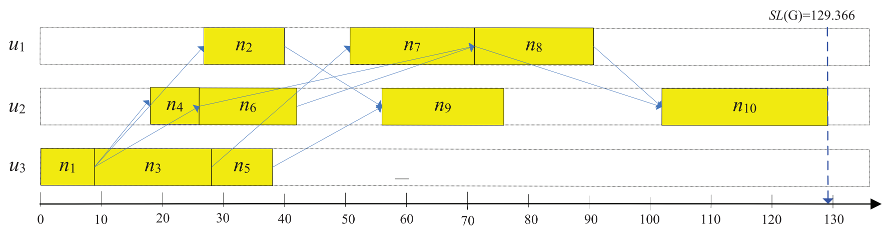

FIGURE 2 Scheduling of the application in Figure 1 using minimum schedule length with energy consumption constraint

follows: 10 ms $\leqslant w_{i,k} \leqslant$ 100 ms, 10 ms $\leqslant c_{i,j} \leqslant$ 100 ms, $0.03 \leqslant P_{k,\text{ind}} \leqslant 0.07$, $0.8 \leqslant C_{k,\text{ef}} \leqslant 1.2$, $2.5 \leqslant m_k \leqslant 3.0$, and $f_{k,\max} = 1$ GHz. All the frequencies are discrete, and the precision is 0.01 GHz. All parallel applications will be executed in a heterogeneous multiprocessor platform with 64 processors. Real parallel applications with precedence-constrained tasks are widely used in high-performance computing, such as FFT and Gaussian elimination.[4] To verify the effectiveness and feasibility of the proposed approach, we use the aforementioned 2 types of real parallel applications to observe the results.

5.2 | FFT parallel applications

A new parameter ρ is used as the size of the FFT parallel application, and the total number of tasks is[4] $|N| = (2 \times \rho - 1) + \rho \times \log_2^{\rho}$, where $\rho = 2^y$ for some integer y. Figure 3 shows an example of the FFT parallel application with $\rho = 8$. Note that ρ exit tasks exist in the FFT application with the size ρ. To adapt the application model of this study, we just add a virtual exit task, and the last ρ tasks are set as the immediate predecessor tasks of the virtual task.

Experiment 1. This experiment is conducted to compare the actual energy consumption values and final schedule lengths of FFT parallel applications with varying energy consumption constraints. We limit the size of the application to $\rho = 32$ (ie, $|N| = 233$). $E_{given}(G)$ is transformed from $(E_{\min}(G) + E_{\max}(G))/10$ into $(E_{\min}(G) + E_{\max}(G))/6$.

As shown in Table 4, the actual energy consumption of the applications using both HEFT and ECS cannot satisfy the individual energy consumption constraints in all the cases. Such results verify that ECS is not designed to satisfy the energy consumption constraints of practical applications. By contrast, MSLECC can always satisfy the energy consumption constraints and the actual energy consumption values are increasingly close to the energy consumption constraints. For example, when $E_{given}(G) = 4486.41$ KJ (1 J = 1 W × 1 s), the energy consumption values using HEFT and ECS are 8852.2 and 6049.71 KJ, respectively, whereas that using MSLECC is 4486.41 KJ, which is considerably close to 4486.41 KJ. In addition, schedule lengths have been effectively controlled within acceptable scopes using MSLECC to satisfy the energy consumption constraints, although the schedule lengths obtained using MSLECC are slightly longer than those using HEFT and ECS in this experiment.

Experiment 2. To observe the performance at different application scales, this experiment is conducted to compare the actual energy consumption values and the final schedule lengths of FFT parallel applications under varying numbers of tasks. We limit $E_{given}(G)$ to $E_{textgiven}(G) = (E_{\min}(G) + E_{\max}(G))/6$. ρ is changed from 8 to 128; that is, the number of tasks is changed from 33 (small scale) to 1151 (large scale).

As shown in Table 5, the energy consumption constraints and actual energy consumption values are increased gradually with the number of tasks. However, the actual energy consumption values of applications using HEFT and ECS still cannot satisfy the energy consumption constraints at different scales. The differences between $E_{given}(G)$ and $E(G)$ increase significantly with the number of tasks. By contrast, MSLECC can always satisfy the energy consumption constraints, and the actual energy consumption values remain close to the energy consumption constraints. For example, when $|N| = 1151$, the energy consumption constraint is $E_{given}(G) = 22632.85$ KJ, but the actual energy consumption values using HEFT and ECS are 38 737.3 and 29 517.43 KJ, respectively, which clearly exceed the given energy consumption constraint. Conversely, the actual energy consumption using MSLECC is 22 598.84 KJ, which is close to the given energy consumption constraint, with a difference of only 34.01 KJ.

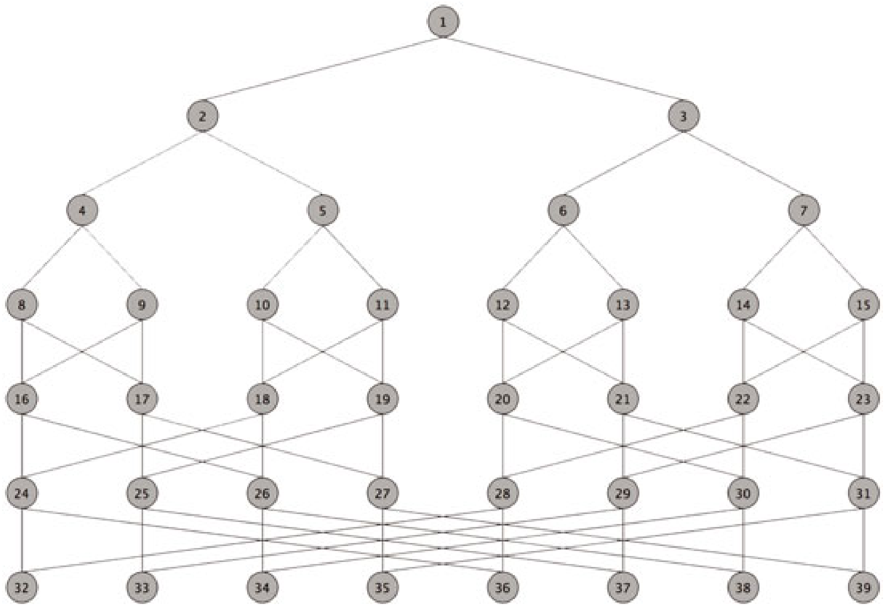

FIGURE 3 Example of the fast Fourier transform parallel application with $\rho = 8$

TABLE 4 Actual energy consumptions (unit: KJ) and final schedule length (unit: ms) of FFT parallel applications with $\rho = 32$ for varying energy consumption constraints

			HEFT[4]		ECS[20]		MSLECC	
$E_{\min}(G)$	$E_{\max}(G)$	$E_{\text{given}}(G)$	*E(G)*	*SL(G)*	*E(G)*	*SL(G)*	*E(G)*	*SL(G)*
654.63	26328.44	2698.30	8809.7	811	5913.40	1055.38	2698.30	1386.57
618.91	26304.68	2991.51	8392.5	893	5928.01	927.12	2991.50	1082.6
622.91	25829.49	3306.55	8057.6	797	5856.44	894.25	3306.53	1051.56
649.26	26372.31	3860.22	8949.7	916	6091.70	1092.67	3860.21	1311.72
629.15	26289.34	4486.41	8852.2	847	6049.71	867.21	4486.41	887.76

Abbreviations: ECS, energy-conscious scheduling; FFT, fast Fourier transform; HEFT, heterogeneous earliest finish time; MSLECC, minimum schedule length with energy consumption constraint.

TABLE 5 Actual energy consumptions (unit: KJ) and final schedule length (unit: ms) of FFT parallel applications for varying number of tasks

					HEFT[4]		ECS[20]		MSLECC	
ρ	$\|N\|$	$E_{\min}(G)$	$E_{\max}(G)$	$E_{\text{given}}(G)$	*E(G)*	*SL(G)*	*E(G)*	*SL(G)*	*E(G)*	*SL(G)*
8	39	112.62	4647.52	793.353	1425.52	459.9	943.11	537.79	793.33	553.84
16	95	264.21	11 324.02	1931.37	3591.45	611.47	2413.73	708.71	1931.36	750.92
32	233	630.03	26 226.91	4476.15	8463.99	859.7	5975.67	925.11	4476.14	979.77
64	511	1442.85	59 923.89	10 227.79	18 685.90	1450.6	14 241.24	1187.3	10 213.94	1162.65
128	1151	3091.14	132 705.98	22 632.85	38 737.37	1950.8	29 517.43	1546.5	22 598.84	1329.0

Abbreviations: ECS, energy-conscious scheduling; FFT, fast Fourier transform; HEFT, heterogeneous earliest finish time; MSLECC, minimum schedule length with energy consumption constraint.

In addition to satisfying the energy consumption constraints, an interesting phenomenon is that MSLECC can also generate shorter schedule lengths than HEFT and ECS in large-scale parallel applications (eg, $|N| = 511$ and $|N| = 1151$). For example, when $|N| = 1151$, the schedule length using MSLECC is 1329 ms, which is considerably less than 1950 and 1546.5 ms using HEFT and ECS, respectively. Such results indicate that lower energy consumption does not cause longer schedule length for large-scale parallel application if a good algorithm can be presented. Then we can draw the following conclusions: (1) MSLECC is highly suitable for minimizing the schedule length of energy consumption-constrained parallel applications and (2) energy consumption optimization is extremely desirable and useful for large-scale parallel applications.

5.3 | Gaussian elimination parallel applications

To further verify the performance of MSLECC, this section uses another important real parallel application, namely, Gaussian elimination, as the experimental object. A new parameter ρ is used as the matrix size of the

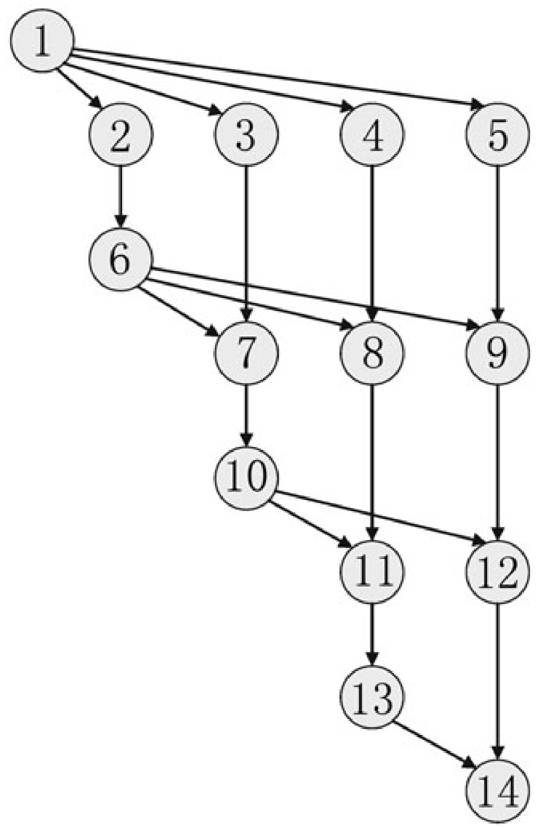

FIGURE 4 Example of the Gaussian elimination parallel application with $\rho = 5$

Gaussian elimination application, and the total number of tasks is[4] $|N| = \frac{\rho^2+\rho-2}{2}$. Figure 4 shows an example of the Gaussian elimination parallel application with $\rho = 5$.

Experiment 3. To observe the results in large-scale cases, similar to that in experiment 2, this experiment is conducted to compare the actual energy consumption values and the final schedule lengths of Gaussian elimination parallel applications under varying numbers of tasks. We limit $E_{\text{given}}(G)$ to $E_{\text{given}}(G) = (E_{\min}(G) + E_{\max}(G))/6$. ρ is changed from 10 to 50; that is, the number of tasks is changed from 54 (small scale) to 1274 (large scale).

Compared with the results in Table 5 for experiment 2, the minimum and maximum energy consumption values of Gaussian elimination parallel applications in Table 6 for experiment 3 are similar to those of FFT parallel applications at the same scale levels. However, Gaussian elimination applications have longer schedule lengths than FFT applications in all the cases. For example, when the task number exceeds 1000, the schedule lengths of the FFT and Gaussian elimination applications using MSLECC are 1329 and 6550.57 ms, respectively. The former is merely one-fifth that of the latter. Such results indicate that FFT applications exhibit better parallelism than Gaussian elimination applications in the structure and can generate shorter schedule lengths.

Similar to the results of the FFT applications in Table 5, the actual energy consumption values of the applications obtained using HEFT and ECS still cannot satisfy the energy consumption constraints in Gaussian elimination at different scales in Table 6. However, the actual energy consumption values obtained using MSLECC satisfy and are close to the energy consumption constraints at different scales. Such results indicate that regardless of the complexity of parallel applications, MSLECC can always satisfy the given energy consumption constraints with the minimum schedule length. Moreover, MSLECC can also generate shorter schedule lengths than HEFT and ECS for large-scale FFT applications (eg, $|N| = 464$, $|N| = 819$, and $|N| = 1274$). Such results further verify that lower energy consumption does not cause longer schedule length for large-scale parallel applications (whether low or high parallelism) if the MSLECC algorithm is used.

From the combined results of the FFT and Gaussian elimination applications, the proposed MSLECC is highly effective in schedule length minimization to satisfy the given energy consumption constraints. We believe that our proposed MSLECC algorithm can effectively improve a section of energy-aware design for parallel applications in heterogeneous distributed environments during the design phase.

6 | CONCLUSIONS

We have developed an effective and low time complexity schedule length minimization algorithm, namely, MSLECC, for energy consumption-constrained parallel applications in heterogeneous distributed systems based on a DVFS energy-efficient design technique. First, our algorithm can always satisfy the energy consumption constraint, and its correctness is verified through proofs and experiments. Second, our MSLECC algorithm implements effective and low time complexity task scheduling to minimize schedule length. Our MSLECC algorithm is highly efficient in satisfying the energy consumption constraint and in minimizing schedule length for large-scale real parallel applications compared with existing energy-efficient algorithms. We believe that our MSLECC algorithm can effectively improve a section

TABLE 6 Actual energy consumptions (unit: KJ) and final schedule length (unit: ms) of Gaussian elimination parallel applications for varying number of tasks

					HEFT[4]		ECS[20]		MSLECC	
ρ	$\|N\|$	$E_{\min}(G)$	$E_{\max}(G)$	$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
10	54	159.57	6403.48	1093.84	2095.94	885	1428.12	928.91	1093.83	1041.73
20	209	561.78	24 345.51	4151.21	8802.86	1923	5812.81	1947.68	3390.34	1967
30	464	909.66	54 798.22	9352.22	19 187.41	4252	14 040.92	3116	9116.86	3013.49
40	819	2223.93	97 185.26	16 568.19	35 433.90	6321	24 915.00	4163.31	16 568.19	3773.51
50	1274	3746.70	148 493.53	25 373.37	57 099.17	7893	40 047.58	6550.57	25 373.35	6316.93

Abbreviations: ECS, energy-conscious scheduling; HEFT, heterogeneous earliest finish time; MSLECC, minimum schedule length with energy consumption constraint.

of energy-aware design for parallel applications in heterogeneous distributed environments during the design phase.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to the anonymous reviewers for their constructive comments that have helped to improve the quality of the manuscript.

REFERENCES

1. Li K. Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers. *IEEE Trans Comput*. 2012;61(12):1668–1681.
2. Khan SU, Zomaya AY. *Handbook on Data Centers*. New York: Springer; 2015.
3. Li K. Energy-efficient and high-performance processing of large-scale parallel applications in data centers. *Handbook on Data Centers*. New York: Springer; 2015:3–35.
4. Topcuoglu H, Hariri S, Wu M-Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans Parallel Distrib Syst*. 2002;13(3):260–274.
5. Khan MA. Scheduling for heterogeneous systems using constrained critical paths. *Parallel Comput*. 2012;38(4):175–193.
6. Xie G, Li R, Li K. Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems. *J Parallel Distrib Comput*. 2015;83: 1–12.
7. Xie G, Liu L, Yang L, Li R. Scheduling trade-off of dynamic multiple parallel workflows on heterogeneous distributed computing systems. *Concurr Comput Pract Experience*. 2016:1–18.
8. Xie G, Zeng G, Liu L, Li R, Li K. High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems. *J Syst Archit*. 2016:1–12.
9. Xie G, Zeng G, Liu L, Li R, Li K. Mixed real-time scheduling of multiple dags-based applications on heterogeneous multi-core processors. *Microprocess Microsyst*. 2016:1–11.
10. Weiser M, Welch B, Demers A, Shenker S. Scheduling for reduced cpu energy. *Mobile Computing*. New York: Springer; 1996:449–471.
11. Mahapatra RN, Zhao W. An energy-efficient slack distribution technique for multimode distributed real-time embedded systems. *IEEE Trans Parallel Distrib Syst*. 2005;16(7):650–662.
12. Zhong X, Xu C-Z. Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee. *IEEE Trans Comput*. 2007;56(3):358–372.
13. Quan G, Hu XS. Energy efficient dvs schedule for fixed-priority real-time systems. *ACM Trans Embedded Comput Syst*. 2007;6(4):150–151.
14. Zhuo J, Chakrabarti C. Energy-efficient dynamic task scheduling algorithms for dvs systems. *ACM Trans Embedded Comput Syst*. 2008;7(2):421–434.
15. Han J-J, Wu X, Zhu D, Jin H, Yang LT, Gaudiot J-L. Synchronization-aware energy management for VFI-based multicore real-time systems. *IEEE Trans Comput*. 2012;61(12):1682–1696.
16. Li K. Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. *IEEE Trans Parallel Distrib Syst*. 2008;19(11):1484–1497.
17. Rusu C, Melhem R, Mossé D. Maximizing rewards for real-time applications with energy constraints. *ACM Trans Embedded Comput Syst (TECS)*. 2003;2(4):537–559.
18. Li K, Tang X, Li K. Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Trans Parallel Distrib Syst*. 2014;25(11):2867–2876.
19. Zong Z, Manzanares A, Ruan X, Qin X. Ead and pebd: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters. *IEEE Trans Comput*. 2011;60(3):360–374.
20. Lee YC, Zomaya AY. Energy conscious scheduling for distributed computing systems under different operating conditions. *IEEE Trans Parallel Distrib Syst*. 2011;22(8):1374–1381.
21. Li K. Power and performance management for parallel computations in clouds and data centers. *J Comput Syst Sci*. 2016;82(2):174–190.
22. Arabnejad H, Barbosa JG. A budget constrained scheduling algorithm for workflow applications. *J Grid Comput*. 2014;12(4):665–679.
23. Zhang L, Li K, Xu Y, Mei J, Zhang F, Li K. Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. *Inf Sci*. 2015;319: 113–131.
24. Huang Q, Su S, Li J, Xu P, Shuang K, Huang X. Enhanced energy-efficient scheduling for parallel applications in cloud. *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, IEEE Computer Society, Ottawa, Canada; 2012:781–786.
25. Tang Z, Qi L, Cheng Z, Li K, Khan SU, Li K. An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. *J Grid Comput*. 2016;14(1):55–74.
26. Zhang J, Li K, Guo D, Qi H, Li W, Jin Y. MDFS: deadline-driven flow scheduling scheme in multi-resource environments. *IEEE Trans Multi-Scale Comput Syst*. 2015;1(4):207–219.
27. Zhu D, Aydin H. Reliability-aware energy management for periodic real-time tasks. *IEEE Trans Comput*. 2009;58(10):1382–1397.
28. Zhao B, Aydin H, Zhu D. Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints. *ACM Trans Des Autom Electron Syst*. 2013;18(2):99–109.

How to cite this article: Xie, G., Xiao, X., Li, R., and Li, K. (2016), Schedule Length Minimization of Parallel Applications with Energy Consumption Constraints using Heuristics on Heterogeneous Distributed Systems, *Concurrency Computat.: Pract. Exper.*, doi:10.1002/cpe.4024.