Journal of Circuits, Systems, and Computers
Vol. 28, No. 11 (2019) 1950190 (23 pages)

DOI: 10.1142/S0218126619501901

World Scientific
www.worldscientific.com

Enhanced Parallel Application Scheduling Algorithm with Energy Consumption Constraint in Heterogeneous Distributed Systems*

Jinghong Li[†,¶], Guoqi Xie[‡,‖], Keqin Li[§,**] and Zhuo Tang[†,††]

[†]*College of Computer Science and Electronic Engineering, Hunan University, P. R. China*

[‡]*Key Laboratory for Embedded and Network Computing of Hunan Province, College of Computer Science and Electronic Engineering, Hunan University, P. R. China*

[§]*Department of Computer Science, State University of New York, USA*
[¶]*li_jinghong@hnu.edu.cn*
[‖]*xgqman@hnu.edu.cn*
[**]*lik@newpaltz.edu*
[††]*tangzhuo@szzt.com.cn*

Received 20 June 2018
Accepted 30 October 2018
Published 14 December 2018

Energy consumption has always been one of the main design problems in heterogeneous distributed systems, whether for large cluster computer systems or small handheld terminal devices. And as energy consumption explodes for complex performance, many efforts and work are focused on minimizing the schedule length of parallel applications that meet the energy consumption constraints currently. In prior studies, a pre-allocation method based on dynamic voltage and frequency scaling (DVFS) technology allocates unassigned tasks with minimal energy consumption. However, this approach does not necessarily result in minimal scheduling length. In this paper, we propose an enhanced scheduling algorithm, which allocates the same energy consumption for each task by selecting a relatively intermediate value among the unequal allocations. Based on the two real-world applications (Fast Fourier transform and Gaussian elimination) and the randomly generated parallel application, experiments show that the proposed algorithm not only achieves better scheduling length while meeting the energy consumption constraints, but also has better performance than the existing parallel algorithms.

Keywords: Heterogeneous distributed systems; energy consumption; DVFS; directed acyclic graph; parallel application; schedule length.

*This paper was recommended by Regional Editor Tongquan Wei.
‖Corresponding author.

1. Introduction

1.1. *Background*

Compared to the response speed of the task and system resource utilization during the execution of parallel application, the energy consumption is also an indicator that is worthy of attention in system design. With the rapid development of hardware technology, the scale and performance of chip integration continue to increase, and modern chips have already reached the level of a few hundred watts. Intel's Itanium2, for example, consumes about 130 watts, and it also requires expensive packaging, heat sinks, and cooling environments, all of which increase energy consumption.[1] According to Moore's Law, chip integration will double every 18 months, but it takes a full 5 years to achieve the corresponding power technology. It reveals that energy consumption has become a problem that must be considered in system development.

In terms of low-power technology, minimizing system energy consumption has always been a goal pursued by researchers.[2] Different technologies are used to achieve this goal. At present, most major chip vendors have implemented these technologies (DPM and DVS/DVFS technology) in their own chips. The basic idea of DPM technology is to save energy consumption by placing the current idle system components in a low-power state when the system is running. And the DVS/DVFS technology is to increase energy efficiency ratio by changing the operating voltage/frequency of the processor.[3]

In terms of task scheduling, how applications can make full use of processors in computer systems for high-performance computing has become a valuable research area. According to a certain scheduling policy, the task scheduling allocates the pre-partitioned sub-tasks to processors in the computer system, so that each sub-task gets the fastest response, thereby shortening the total execution time of the entire task set, and balance system load and optimize system operation efficiency.[4,5]

In general, parallel application scheduling problems are NP-hard.[6] Applications are represented by directed acyclic graph (DAG) and are widely used for static scheduling problems, similar to the work of Braun,[7] where nodes represent application tasks and edges denote data dependencies between tasks. Therefore, we also use the DAG model to represent parallel applications.

1.2. *Motivation*

With the increasing use of multiprocessors in high-performance embedded systems, a series of applications, such as image recognition, human body interaction and so on, are occuring. In Ref. 8, a fast functional safety verification(FFSV) method is provided for the early design phase of distributed automotive applications. Xie *et al.*[9] proposed a dynamic scheduling algorithm based on fairness (FDS_MIMF) and an adaptive dynamic scheduling algorithm (ADS_MIMF), which can automatically

meet the challenges of heterogeneity, dynamics and parallelism of automotive cyber-physical systems (ACPS).[10] Since embedded systems are cost-sensitive, hardware costs and system resource consumption costs are reduced as much as possible when functional security requirements are met. The above works ignore the effect of the elevated operating temperature in the system.

In Ref. 11, Zhou *et al.* studied algorithms for optimizing the completion time under the constraints of reliability and temperature. Due to the increase of chip temperature, energy-saving task scheduling with thermal considerations has also become a research topic for many researchers. In Ref. 12, the proposed random thermal sensing task scheduling algorithm takes into account the uncertainty of the instantaneous fault occurrences. In Ref. 13, a two-stage energy-efficient temperature-aware task scheduling scheme for heterogeneous real-time MPSoC systems is designed. Wei *et al.*[14] used approximate calculations to intelligently handle the uncertainty of energy availability with limited energy.

Therefore, we need to find a balance between energy consumption and completion time. The MSLECC algorithm proposed in Ref. 15 solves the minimum scheduling problem of parallel application under energy constraints. Since the algorithm is not very satisfactory, a more efficient scheduling algorithm is needed.

1.3. *Our contributions*

In this paper, an enhanced scheduling algorithm (EECC) is proposed for parallel applications in heterogeneous distributed computing systems. Our goal is to minimize the schedule length while the energy consumption constraints of parallel applications is satisfied. The experimental results also show that the algorithm can achieve better schedule length while satisfying the energy consumption constraints.

The main contributions of this paper can be summarized as follows:

- We propose a new task scheduling algorithm to schedule parallel applications, which can obtain a better scheduling length while still meeting energy consumption constraints, and achieve higher performance with lower time complexity.
- We use two real-world applications and the randomly generated parallel application to verify the effectiveness of the proposed EECC algorithm. The results show that under different conditions, the algorithm can obtain better schedule length compared to other parallel algorithms.

The rest of the paper is organized as follows. We compare our work with related research works in Sec. 2. Section 3 describes the application, energy models and preliminaries used in this paper. Section 4 presents the detail of the problem and our scheduling algorithm EECC. Results from experimental evaluation are reported in Sec. 5. We conclude this paper in Sec. 6.

2. Related Works

In recent years, many scheduling strategies[11,16–18] have focused on saving energy on a single processor, or on a homogeneous multiprocessor system or in heterogeneous resources. Currently, many effective technologies have been studied to reduce energy consumption, such as the DVFS[19,20] mentioned in Sec. 1. Based on this, a large number of task scheduling works have been proposed, and slack time reclamation technique has also been used in many recent studies. Kim *et al.*[17] provided a power-aware scheduling algorithm with deadline-constrained bag-of-tasks applications on DVS-enabled cluster systems. In Ref. 21, by using non-DVFS and global DVFS energy efficiency scheduling algorithms, the problem of minimizing energy consumption while satisfying deadline constraints in real-time parallel applications on heterogeneous distributed systems is solved. In Ref. 22, two novel scheduling algorithms for a limited number of heterogeneous processors are proposed, the goal of which is to simultaneously satisfy high performance and fast scheduling time.

Two energy-conscious scheduling heuristic algorithms were proposed by Lee *et al.*,[23] they are considering the makespan of parallel tasks in heterogeneous distributed computing systems while also considering the energy consumption. In Ref. 18, Xie *et al.* not only maximized the number of workflows completed within the deadline, but also minimized the energy consumption of workflows completed during the deadline. In Ref. 24, Huang *et al.* presented an enhanced energy-efficient scheduling (EES) algorithm that globally analyzes and utilizes idle space to minimize energy consumption while still meeting the deadline of heterogeneous computing systems. However, this strategy only minimizes energy consumption through the upward approach, while in Ref. 25, the downward and upward approaches solved the same problem. The above works are to minimize the tasks' energy consumption while meeting the service level agreement (SLA) based on performance.

Nevertheless, in addition to the above works, most other studies only focused on shortening the makespan or reducing energy consumption, or reducing the dispatch length rather than the energy consumption. Different from the above studies, Xiao *et al.*[15] proposed an algorithm (MSLECC), which solves the problem of minimizing the parallel applications' dispatch length with limited energy consumption in heterogeneous distributed systems based on DVFS technology. Although the algorithm achieved a minimum scheduling length while meeting the constrained energy consumption, the result is not ideal. In this study, we propose an enhanced scheduling algorithm to solve the same problem.

3. Models and Preliminaries

3.1. *Application model*

In this study, we use $U = \{u_1, u_2, \ldots, u_{|U|}\}$ to denote a set of heterogeneous processors, where $|U|$ indicates the size of set U. Generally, parallel applications can

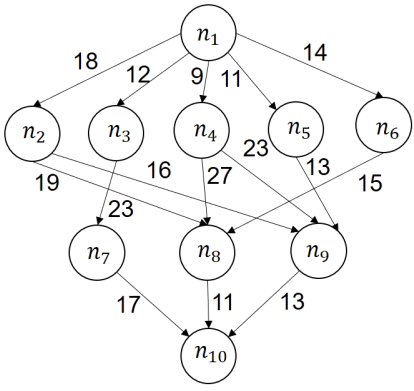


Fig. 1. Standard example of a DAG-based parallel application.

be represented by a directed acyclic graph (DAG) $G = (N, M, W, C)$[8,22,26] as shown in Fig. 1. N indicates a set of nodes in G, and each node denotes a task performed on different processors with different execution times. M is an edge set describing the data dependencies between tasks in the execution of the parallel application, and each edge $m_{i,j} \in M$ is connected to two nodes n_i and n_j. W is a $|N| \times |U|$ matrix where $w_{i,k}$ indicates the time required for task n_i to execute on processor u_k with maximum frequency. Accordingly, $c_{i,j} \in C$ represents the communication time of $m_{i,j}$ while n_i and n_j are not assigned to the same processor. However, if the task n_i and n_j are both on the same processor, the communication cost c_{n_i,n_j} will be assumed to be zero. In DAG diagram, the set of the immediate predecessor tasks of n_i is represented by $pred(n_i)$, and the set of its immediate successor tasks is represented by $succ(n_i)$. If a task has no predecessor task, we call it an entry task n_{entry}; and a task without any successor task is called an exit task n_{exit}.

As shown in Fig. 1, a DAG-based parallel application consists of ten tasks which performed on three processors $\{u_1, u_2, u_3\}$. In this DAG, when n_1 and n_3 are not assigned to the same processor, the weight 12 of the edge between n_1 and n_3 indicates the communication time, which is represented by $c_{1,3}$. And as shown in Table 1, the execution time of task n_1 on processor u_1 is 14 with the maximum frequency. Due to the heterogeneity of the processors, we can see from Table 1 that the same task is executed on different processors with different times.

Table 1. Execution time values of tasks on different processors with the maximum frequencies of the application in Fig. 1.

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

3.2. *Energy model*

Due to the almost linear relationship between supply voltage and operating frequency, DVFS saves power/energy by reducing the supply voltage and clock frequency. In this work, we adopt the system-level power model originally proposed in Refs. 27–29. Therefore, the power consumption $P(f)$ of a computing system with frequency f is given by

$$P(f) = P_{\mathrm{s}} + h(P_{\mathrm{ind}} + P_{\mathrm{d}}) = P_{\mathrm{s}} + h(P_{\mathrm{ind}} + C_{\mathrm{ef}} f^m)\,. \tag{1}$$

Above P_{s} represents static power, which is usually used to maintain the basic circuits and keep the clock running. It can only be eliminated by turning off the entire system. P_{ind} is a constant that represents the frequency-independent dynamic power and corresponds to the power independent of the CPU processing speed. P_{d} denotes the frequency-dependent dynamic power, including the power primarily consumed by the CPU and any power that depends on the system processing frequency f. h represents the system states, specifically, when the computation is in progress, the system is active, $h = 1$; otherwise, $h = 0$, indicating that the system is in a power-saving sleep mode or off. C_{ef} denotes the effective switching capacitance, and m denotes the dynamic power index (generally not less than 2), which are all system-dependent constants.

Considering the excessive time and energy overhead associated with turning on/off the system, we will ignore the static power due to the unmanageability of P_{s} and concentrate our analysis on P_{ind} and P_{d} in this paper. Although DVFS can reduce energy consumption, application require more time to complete at low frequencies. Therefore, given the system-level power, lower frequencies may not always be optimal for energy savings. Thus, the minimum energy-efficient frequency f_{ee} exists,[27–29] and it is expressed as

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

Assuming that the frequency of the processor can be changed continuously between $f_{\min}$, the minimum available frequency, and $f_{\max}$, the maximum frequency. Consequently, for energy efficiency, the actual effective frequency f should be limited to the range $[f_{\mathrm{low}}, f_{\max}]$ where $f_{\mathrm{low}} = \max(f_{\min}, f_{\mathrm{ee}})$. Each processor should has separate parameters, due to the heterogeneous of processors. So we define the frequency-independent dynamic power set $\{P_{1,\mathrm{ind}}, P_{2,\mathrm{ind}}, \ldots, P_{|U|,\mathrm{ind}}\}$, the frequency-dependent dynamic power set $\{P_{1,\mathrm{d}}, P_{2,\mathrm{d}}, \ldots, P_{|U|,\mathrm{d}}\}$, the effective switching capacitance set $\{C_{1,\mathrm{ef}}, C_{2,\mathrm{ef}}, \ldots, C_{|U|,\mathrm{ef}}\}$, the dynamic power exponent set $\{m_1, m_2, \ldots, m_{|U|}\}$, the minimum energy-efficient frequency set $\{f_{1,\mathrm{ee}}, f_{2,\mathrm{ee}}, \ldots, f_{|U|,\mathrm{ee}}\}$, and the actual effective frequency set

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

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

$$E(n_i, u_k, f_{k,h}) = (P_{\text{ind}} + C_{\text{k,ef}} \times (f_{k,h})^{m_k}) \times w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{3}$$

3.3. *Preliminaries*

(1) Earliest start time (EST), earliest finish time (EFT)
$\text{EST}(n_i, u_k, f_{k,h})$ refers to the earliest start time of the task n_i with the frequency $f_{k,h}$ on the processor u_k, and EFT refers to the earliest finish time. EFT is considered the standard for task assignment, so each task chooses the minimum EFT to achieve the applications current shortest scheduling length.

For an entry task, its earliest start time (EST) on any processor is zero. For other tasks in the DAG diagram, the EST and EFT attribute values are obtained by iterative calculating from the entry task. The above calculation methods are:

$$\text{EST}(n_i, u_k, f_{k,h}) = \max_{n_x \in pred(n_i)} \{avail[k], \max\{AFT(n_x) + c'_{x,i}\}\}, \tag{4}$$

$$\text{EFT}(n_i, u_k, f_{k,h}) = \text{EST}(n_i, u_k, f_{k,h}) + w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{5}$$

$avail[k]$ is the earliest time that the processor u_k can execute the task n_i; $AFT(n_x)$ indicates the actual finish time of task n_x; and $c'_{x,i}$ indicates the actual communication time between tasks n_x and n_i.

(2) Schedule length (SL)
After all the tasks in the DAG map have been scheduled, the scheduling length of the algorithm is also determined, and the value is equal to the actual finish time of the exit task, that is:

$$SL(G) = AFT(n_{\text{exit}}).$$

(3) Upward rank value
Since the rank is calculated by traversing the entire application from the exit task, it is called an "upward rank". Similar to most literature, we will use the upward rank value (rank_u)[20–22] of the task given in Eq. (6) to sort all tasks in descending order.

$$\text{rank}_{\text{u}}(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{c_{i,j} + \text{rank}_{\text{u}}(n_j)\}, \tag{6}$$

where $\overline{w_i}$ indicates the average execution time of task n_i, and it can be calculated as $\overline{w_i} = (\sum_{k=1}^{|U|} w_{i,k})/|U|$. Table 2 shows the upward rank values of all the tasks in Fig. 1.

(4) Energy consumption constraint
Since the available frequency on each processor and the execution time of each task are known, the minimum and maximum energy consumption of each task on a

Table 2. Upward rank values for tasks of the motivating parallel application in Fig. 1.

Task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$\text{rank}_\text{u}(n_i)$	108	77	80	80	69	63.33	42.67	35.67	44.33	14.67

certain processor can be obtained, expressed as $E_{\min}(n_i)$ and $E_{\max}(n_i)$ respectively. Its equations are denoted by

$$E_{\min}(n_i) = \min_{u_k \in U} E(n_i, u_k, f_{k,\text{low}}) \,, \tag{7}$$

$$E_{\max}(n_i) = \max_{u_k \in U} E(n_i, u_k, f_{k,\max}) \,, \tag{8}$$

respectively.

Thus, the minimum and maximum energy consumption of the application G can be obtained. The calculation equations are

$$E_{\min}(G) = \sum_{i=1}^{|N|} E_{\min}(n_i) \,, \tag{9}$$

$$E_{\max}(G) = \sum_{i=1}^{|N|} E_{\max}(n_i) \,. \tag{10}$$

In this paper, we define $E_{\text{given}}(G)$ as the given energy consumption constraint the application must be satisfied, and it is limited to the range $[E_{\min}(G), E_{\max}(G)]$. If $E_{\text{given}}(G) < E_{\min}(G)$, $E_{\text{given}}(G)$ is always not satisfied; if $E_{\text{given}}(G) > E_{\max}(G)$, $E_{\text{given}}(G)$ is always satisfied, this will make no sense.

4. Enhanced Energy Consumption Constrained Scheduling

4.1. *Problem definition*

In this study, the problem to be solved is to allocate available processors with the appropriate frequency for all tasks to minimize the application's schedule length, while ensuring that the application's energy consumption does not exceed it's energy constraints. The objective is expressed in expression as

$$E(G) = \sum_{i=1}^{|N|} E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leq E_{\text{given}}(G) \,, \tag{11}$$

where $u_{pr(i)}$ and $f_{pr(i),hz(i)}$ indicate the allocated processor and frequency of n_i, respectively, and $f_{pr(i),\text{low}} \leq f_{pr(i),hz(i)} \leq f_{pr(i),\max}$, for $\forall i : 1 \leq i \leq |N|, u_{pr(i)} \in U$.

4.2. *Satisfying energy consumption constraint*

We schedule the tasks based on the upward rank values. Suppose that the task currently to be allocated is $n_{\text{ord}(j)}$, then the task set where the tasks have been

allocated is $\{n_{\text{ord}(1)}, n_{\text{ord}(2)}, \ldots, n_{\text{ord}(j-1)}\}$, and the task set where the tasks have not be allocated is $\{n_{\text{ord}(j+1)}, n_{\text{ord}(j+2)}, \ldots, n_{\text{ord}(|N|)}\}$. Previous studies have assumed that each task in $\{n_{\text{ord}(j+1)}, n_{\text{ord}(j+2)}, \ldots, n_{\text{ord}(|N|)}\}$ is allocated to the processor and frequency with minimal energy consumption to ensure that each task assignment met the energy consumption constraints of the application with a minimum sum of energy consumption. Although this approach satisfied energy consumption constraints, it is too passive. Therefore, in this paper we propose an enhanced algorithm, by selecting a relatively intermediate value in the unequal allocation to reduce the schedule length. First of all, we explain the uneven distribution.

We suppose that the first task to be assigned is allocated to the processor with the energy consumption of $E_{\min}(n_{\text{ord}(1)}) + \delta$, the second task to be allocated is $E_{\min}(n_{\text{ord}(2)}) + 2\delta$, and so on with different value of δ. Then the nth task to be distributed is allocated with the energy consumption of $E_{\min}(n_{\text{ord}(n)}) + n\delta$,

$$\delta = \frac{2}{|N|^2 + |N|} \left(E_{\text{given}}(G) - E_{\min}(G)\right). \tag{12}$$

Therefore, the average assignable energy value for each task we select can be calculated as

$$E_\delta(n_i) = E_{\min}(n_i) + \delta \times \frac{|N|}{2}. \tag{13}$$

Correspondingly, the pre-allocated energy of the unassigned task n_i can be expressed as

$$E_{\text{pre}}(n_i) = \min\{E_\delta(n_i), E_{\max}(n_i)\}. \tag{14}$$

Thoerem 1. When assigning the task $n_{\text{ord}(j)}$, the application's energy consumption should satisfy:

$$\begin{aligned} E_{\text{ord}(j)}(G) = & \sum_{x=1}^{j-1} E(n_{\text{ord}(x)}, u_{pr(\text{ord}(x))}, f_{pr(\text{ord}(x)),hz(\text{ord}(x))}) \\ & + E(n_{\text{ord}(j)}, u_k, f_{k,h}) + \sum_{x=j+1}^{|N|} E_{\text{pre}}(n_{\text{ord}(x)}) \le E_{\text{given}}(G). \end{aligned} \tag{15}$$

Proof. The restriction condition expressed by Eq. (15) is proved by mathematical induction. Firstly, when $j = 1$ (i.e., for entry task $n_{\text{ord}(1)}$), the application should meet the following constraint:

$$E_{\text{ord}(1)}(G) = E(n_{\text{ord}(1)}, u_k, f_{k,h}) + \sum_{x=2}^{|N|} E_{\text{pre}}(n_{\text{ord}(x)}) \le E_{\text{given}}(G). \tag{16}$$

According to Eqs. (12)–(14), there is

$$\begin{aligned} E_{\mathrm{ord}(1)}(G) &= E(n_{\mathrm{ord}(1)}, u_k, f_{k,h}) + \sum_{x=2}^{|N|} E_{\mathrm{pre}}(n_{\mathrm{ord}(x)}) \\ &\le E(n_{\mathrm{ord}(1)}, u_k, f_{k,h}) + \sum_{x=2}^{|N|} E_{\delta}(n_{\mathrm{ord}(x)}) \\ &= E(n_{\mathrm{ord}(1)}, u_k, f_{k,h}) + \sum_{x=1}^{|N|} E_{\delta}(n_{\mathrm{ord}(x)}) - E_{\delta}(n_{\mathrm{ord}(1)}) \\ &= E(n_{\mathrm{ord}(1)}, u_k, f_{k,h}) + E_{\mathrm{given}}(G) - (E_{\min}(n_{\mathrm{ord}(1)}) + n\delta)\,. \end{aligned} \tag{17}$$

Obviously, $E_{\min}(n_{\mathrm{ord}(1)}) + n\delta$ is greater than or equal to $E_{\min}(n_{\mathrm{ord}(1)})$, so the processor with the lowest energy consumption can be allocated to $n_{\mathrm{ord}(1)}$ at least.

Therefore, $n_{\mathrm{ord}(1)}$ can find an allocated processor and frequency to satisfy:

$$\begin{aligned} E_{\mathrm{ord}(1)}(G) &= E(n_{\mathrm{ord}(1)}, u_k, f_{k,h}) + E_{\mathrm{given}}(G) - (E_{\min}(n_{\mathrm{ord}(1)}) + n\delta) \\ &\le E_{\mathrm{given}}(G). \end{aligned} \tag{18}$$

Secondly, suppose that the jth task $n_{\mathrm{ord}(j)}$ can search an allocated processor and frequency to meet the constraint, and there have

$$\begin{aligned} E_{\mathrm{ord}(j)}(G) = &\sum_{x=1}^{j-1} E(n_{\mathrm{ord}(x)}, u_{pr(\mathrm{ord}(x))}, f_{pr(\mathrm{ord}(x)),hz(\mathrm{ord}(x))}) \\ &+ E(n_{\mathrm{ord}(j)}, u_{pr(\mathrm{ord}(j))}, f_{pr(\mathrm{ord}(j)),hz(\mathrm{ord}(j))}) \\ &+ \sum_{x=j+1}^{|N|} E_{\mathrm{pre}}(n_{\mathrm{ord}(x)}) \le E_{\mathrm{given}}(G)\,. \end{aligned} \tag{19}$$

Therefore, for the $(j+1)$th task $n_{\mathrm{ord}(j+1)}$, the energy consumption of the application G is

$$\begin{aligned} E_{\mathrm{ord}(j+1)}(G) = &\sum_{x=1}^{j} E(n_{\mathrm{ord}(x)}, u_{pr(\mathrm{ord}(x))}, f_{pr(\mathrm{ord}(x)),hz(\mathrm{ord}(x))}) \\ &+ E(n_{\mathrm{ord}(j+1)}, u_k, f_{k,h}) + \sum_{x=j+2}^{|N|} E_{\mathrm{pre}}(n_{\mathrm{ord}(i)})\,. \end{aligned} \tag{20}$$

According to Eqs. (19) and (20), there is

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

$$\leq E_{\text{given}}(G) - \sum_{x=j+1}^{|N|} E_{\text{pre}}(n_{\text{ord}(x)}) + E(n_{\text{ord}(j+1)}, u_k, f_{k,h}) + \sum_{x=j+2}^{|N|} E_{\text{pre}}(n_{\text{ord}(x)}).$$

Then,

$$E_{\text{ord}(j+1)}(G) \leq E_{\text{given}}(G) + E(n_{\text{ord}(j+1)}, u_k, f_{k,h}) - E_{\text{pre}}(n_{\text{ord}(j+1)}).$$

Since the value of $E_{\text{pre}}(n_{\text{ord}(j+1)})$ is larger than or equal to $E_{\min}(n_{\text{ord}(j+1)})$, we can get $E_{\text{ord}(j+1)}(G) \leq E_{\text{given}}(G)$ similar to the case of $j = 1$. □

This shows that $n_{\text{ord}(G)}$ also satisfies the energy consumption constraint. From the above, we can see that each task can find separate allocated processors and frequencies to meet the energy consumption constraint.

4.3. *EECC algorithm design*

Before explaining the details in this section, we firstly give the energy consumption constraint for each task. According to Eq. (15), there have

$$E_{\text{ord}(j+1)}(G) \leq E_{\text{given}}(G) - \sum_{x=1}^{j-1} E(n_{\text{ord}(x)}, u_{pr(\text{ord}(x))}, f_{pr(\text{ord}(x)),hz(\text{ord}(x))}) - \sum_{x=j+1}^{|N|} E_{\text{pre}}(n_{\text{ord}(x)}).$$

Hence, let the energy consumption constraint of the task be

$$E_{\text{given}}(n_{\text{ord}(j)}) = E_{\text{given}}(G) - \sum_{x=1}^{j-1} E(n_{\text{ord}(x)}, u_{pr(\text{ord}(x))}, f_{pr(\text{ord}(x)),hz(\text{ord}(x))}) - \sum_{x=j+1}^{|N|} E_{\text{pre}}(n_{\text{ord}(x)}). \quad (21)$$

So, when assigning a single task, we only need to consider the energy consumption constraint of the task without considering the application's energy consumption constraint. The main idea of the operation is to convert the application's energy consumption constraint to that of each task. Since the maximum energy consumption constraint of the task $n_{\text{ord}(j)}$ is $E_{\max}(n_{\text{ord}(j)})$, $E_{\text{given}}(n_{\text{ord}(j)})$ should satisfy the following constraint:

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

Algorithm 1. The EECC Algorithm

Input: G: A DAG graph; U: A set of DVFS-enabled processors.
Output: $SL(G)$: the schedule length of the application; $E(G)$: the actual energy consumption of the application.

1: Sort the tasks in a list *downtask_list* by descending order of $rank_{\mathrm{u}}$ values.
2: **while** (tasks in *downtask_list*) **do**
3: $n_i = downtask\_list.out()$;
4: Calculate $E_{\min}(n_i)$ and $E_{\max}(n_i)$ using Eqs. (7), and (8), respectively;
5: Calculate $E_{\min}(G)$ and $E_{\max}(G)$ using Eqs. (9), and (10), respectively;
6: Calculate $E_{\mathrm{pre}}(n_i)$ using Eqs. (12), (13) and (14);
7: Calculate $E_{\mathrm{given}}(n_i)$ using Eq. (??);
8: **for** $(\forall k, U_k \in U)$ **do**
9: **for** $(\forall h, f_{k,h} \in [f_{k,\mathrm{low}}, f_{k,\max}])$ **do**
10: Calculate $E(n_i, u_k, f_{k,h})$ using Eq. (3);
11: **if** $(E(n_i, u_k, f_{k,h}) \leq \min\{E_{\mathrm{given}}(n_i), E_{\max}(n_i)\})$ **then**
12: $f_{k,hz(i)} \leftarrow f_{k,h}$;
13: **end if**
14: **end for**
15: Calculate $EFT(n_i, u_k, f_{k,h})$ using Eq. (5);
16: **if** $(EFT(n_i, u_k, f_{k,h}) < AFT(n_i))$ **then**
17: $pr(i) \leftarrow k$;
18: $f_{pr(i),hz(i)} \leftarrow f_{k,hz(i)}$;
19: $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leftarrow E(n_i, u_k, f_{k,hz(i)})$;
20: $AFT(n_i) \leftarrow EFT(n_i, u_k, f_{k,hz(i)})$;
21: **end if**
22: **end for**
23: **end while**
24: Calculate $E(G)$ using Eq. (11);
25: Calculate $SL(G) = AFT(n_{\mathrm{exit}})$.

Therefore, a enhanced energy consumption constraint algorithm (EECC) is proposed in Algorithm 1. In EECC, only processor and frequency with the minimum EFT is selected for each task in the case of energy consumption constraints. EECC uses the insertion-based scheduling strategy to decrease the schedule length while meeting the energy consumption constraints. Each phase is explained in detail as follows.

(1) In Line 6, the pre-allocated energy of each task is calculated.
(2) In Lines 8–22, By traversing all processors and frequencies, each task selects the processor with the minimum EFT when the energy consumption constraint is satisfied. The time complexity is $O(|N| \times |U| \times |F|)$, where $|F|$ denotes the maximum number of discrete frequencies from the lowest to the largest actual effective frequencies.
(3) In Lines 24 and 25, we calculate the actual energy consumption $E(G)$ and the schedule length $SL(G)$.

Through the above analysis, we can see that EECC algorithm can achieve low time complexity $O(|N|^2 \times |U| \times |F|)$ for parallel applications which have the energy consumption constraints.

Table 3. Processors' power and frequency parameters.

u_k	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	m_k	$f_{k,\text{low}}(f_{k,\text{ee}})$	$f_{k,\max}$
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

Table 4. Task assignment for the application generated by EECC in Fig. 1.

n_i	$E_{\text{given}}(n_i)$	$u(n_i)$	$f(n_i)$	$AST(n_i)$	$AFT(n_i)$	$E(n_i)$
n_1	13.5149	u_3	1.0	0	9	9.63
n_3	11.3514	u_1	1.0	21	32	9.13
n_4	9.8173	u_2	1.0	18	26	6.72
n_2	10.9183	u_3	0.62	9	38.0323	10.8197
n_5	7.7425	u_2	0.77	26	42.8831	7.7023
n_6	7.8612	u_1	0.83	32	47.6627	7.7692
n_9	8.7275	u_2	0.89	54.0323	67.5154	8.5997
n_7	6.8852	u_1	1.0	47.6627	54.6627	5.81
n_8	7.4782	u_1	1.0	57.0323	62.0323	4.15
n_{10}	10.6641	u_2	1.0	73.0323	80.0323	5.88
$E(G) = 76.2109, SL(G) = 80.0323$						

4.4. *Motivational example*

Using Fig. 1 as an example, Table 3 lists all the processors parameters, such as the frequency-independent dynamic power $P_{k,\text{ind}}$, the effective switching capacitance $C_{k,\text{ef}}$ and the dynamic power exponent m_k. Each processor's maximum frequency $f_{k,\max}$ is 1.0 and its frequency precision is 0.01. In this example, the minimum energy-efficient frequency $f_{k,\text{ee}}$ derived from Eq. (2) is considered as $f_{k,\text{low}}$.

Therefore, the minimum and maximum energy consumption of application can be calculated as $E_{\min}(G) = 20.31$ and $E_{\max}(G) = 161.99$ according to Eqs. (9) and (10), respectively. We set the energy constraint of application G as $E_{\text{given}}(G) = 0.5 \times E_{\max}(G)$. Then the task assignment for the parallel application in Fig. 1 are shown in Table 4. Each row represents a task allocation and its relevant values. The actual energy consumption of the application is 76.2109, which is less than $E_{\text{given}}(G)$. And the schedule length is 80.0323.

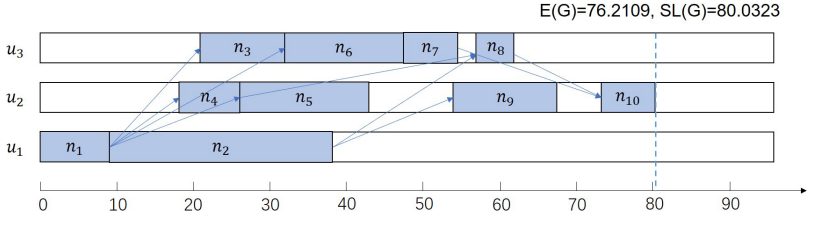


Fig. 2. Scheduling of the application in Fig. 1 using the EECC.

Figure 2 shows a scheduling diagram of the parallel application G in Fig. 1 using the EECC. The arrows in Fig. 2 indicate the communication information generated between tasks.

5. Experiments

In this section, we use two algorithms, HEFT (a well-known algorithm that does not consider energy costs, but performs well in task scheduling[23]) and minimum schedule length with energy consumption constraint algorithm (MSLECC), which are the same as the goal of this paper and propose a comparative evaluation of our algorithm (EECC) to evaluate the performance of our proposed method. We implement a Java simulation platform to validate our algorithm.

The experiment comparisons of the algorithm are based on the following two performance metrics: the actual energy consumption $E(G)$ and the final schedule length $SL(G)$.

The parameters of the processors and applications as follows: $10\,\text{ms} \leq w_{i,k} \leq 100\,\text{ms}$, $10\,\text{ms} \leq c_{i,j} \leq 100\,\text{ms}$, $0.03 \leq P_{k,\text{ind}} \leq 0.07$, $0.8 \leq C_{k,\text{ef}} \leq 1.2$, $2.5 \leq m_k \leq 3.0$, and $f_{k,\max} = 1\,\text{GHz}$. All frequencies are discrete, and the precision is 0.01 GHz. We chose three DAG models to evaluate our algorithm: two real-world applications (Fast Fourier transform and Gaussian elimination) and randomly generated applications.[22]

5.1. *Fast Fourier transform application*

We first consider the fast Fourier transform (FFT), Fig. 3 shows an example of the FFT parallel application with $\rho = 4$. We denote ρ as the application's matrix dimension. The total number of tasks in a fast Fourier transform graph is equal to $(2 \times \rho - 1) + \rho \times \log_2 \rho$, where $\rho = 2^y$ for a certain integer y. Note that there are ρ exit tasks exist in a FFT application with the size of ρ. To adapt this research

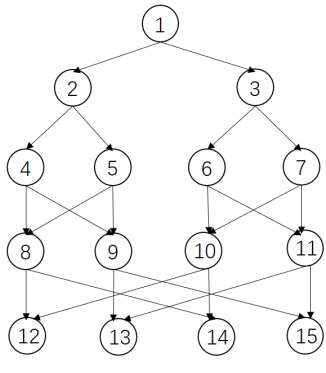


Fig. 3. Example of FFT parallel application with $\rho = 4$.

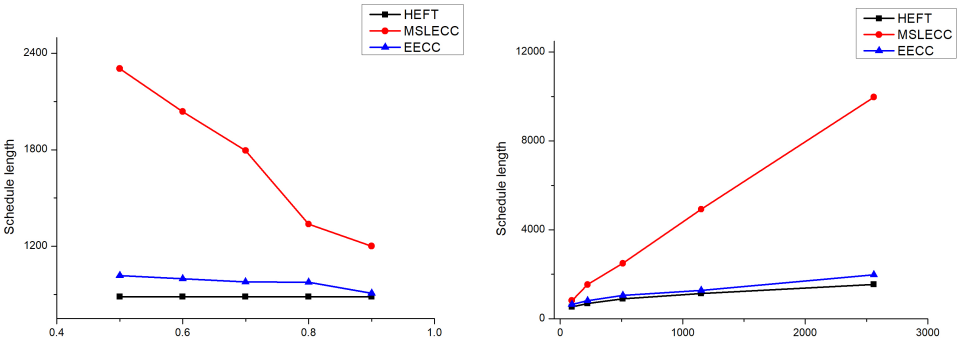
Table 5. Results of FFT parallel applications with $\rho = 64$ for varying $E_{\text{given}}(G)$.

	HEFT		MSLECC		EECC	
$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
5308.12	10616.24	886	5308.12	2305.56	5308.06	1016.64
6369.74	10616.24	886	6369.74	2038.23	6369.59	997.32
7431.37	10616.24	886	7431.37	1795.70	7431.12	976.83
8492.99	10616.24	886	8492.99	1337.65	8471.93	976.00
9554.62	10616.24	886	9554.62	1200.83	9424.86	906.23

application model, we introduce a virtual exit task to connect these tasks, that is, the last ρ tasks are set as the immediate predecessor tasks of the virtual task. Note that the virtual exit task has zero time overhead.

Experiment 1. In order to observe the performance on different energy consumption constraints, an experiment is carried out to compare the actual energy consumption and the final schedule length values of the FFT application for varying energy consumption constraints. We limit the matrix dimension as $\rho = 64$ (i.e., $|N| = 511$), and the energy consumption constraints is changed from $E_{\text{HEFT}}(G) \times 0.5$ to $E_{\text{HEFT}}(G) \times 0.9$. The $E_{\text{HEFT}}(G)$ represents the energy consumption generated by HEFT.

Table 5 shows the details of the final schedule lengths and energy consumption values of fast Fourier transform application with $\rho = 64$ for varying $E_{\text{given}}(G)$ by using all the algorithms, and a more intuitive feeling can be performed through Fig. 4(a). In the three algorithms, although the MSLECC and EECC algorithms can meet the energy consumption constraints in all cases, the EECC algorithm is more effective than the MSLECC in the schedule length. The schedule length basically decreased from 24% to 50%. For example, when $E_{\text{given}}(G) = 5308.12$, the schedule length using EECC is 1016.64 while the schedule length using MSLECC is 2305.56.



(a) Varying $E_{\text{given}}(G)$ (b) Varying number of tasks

Fig. 4. Final schedule length of FFT application.

Table 6. Results of FFT parallel applications for varying number of tasks.

			HEFT		MSLECC		EECC	
ρ	$\|N\|$	$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
16	95	1077.01	2154.01	537	1077.01	819.59	1076.84	646.44
32	223	2354.95	4709.90	692	2354.95	1536.02	2354.90	807.96
64	511	5225.56	10451.11	890	5225.55	2491.77	5225.52	1049.85
128	1151	11577.39	23154.78	1134	11577.39	4924.77	11577.21	1269.53
256	2559	21192.23	42384.46	1540	21192.23	9972.54	21192.22	1980.66

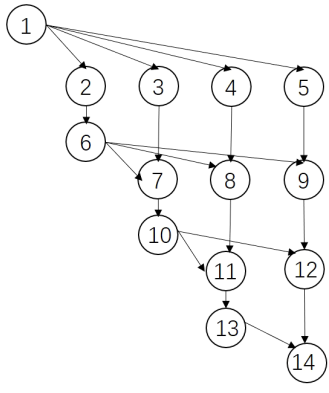


Fig. 5. Example of GE parallel application with $\rho = 5$.

These results indicate that the proper allocation of energy consumption can achieve a better schedule length.

Experiment 2. In order to observe the algorithm performance under different number of tasks, an experiment is carried out to compare the actual energy consumption and the final schedule length values of the FFT application for varying number of tasks. The matrix dimension ρ is changed from 16 to 256, that is, the scale of tasks is changed from 95 to 2559. $E_{\text{given}}(G)$ is set to $E_{\text{HEFT}}(G) \times 0.5$.

Table 6 shows the results of fast Fourier transform applications for different number of tasks by using the three algorithms. It can be seen from the table and Fig. 4(b) that, as the application increases, although the MSLECC and EECC can always meet the energy consumption constraints, the MSLECC is more pessimistic than the EECC-generated schedule length. The actual energy consumption using HEFT applications still cannot meet the energy constraints in different scales. The schedule length basically decreased from 21% to 80%. For example, when $\rho = 128$ (i.e., $|N| = 1151$), the scheduling length using MSLECC is 4924.77 while EECC is 1269.53. And the actual energy consumption using HEFT is 23154.78, which obviously does not meet the given energy consumption constraints 11577.39. These results show that the proposed EECC algorithm has better performance than MSLECC.

Table 7. Results of GE application with $\rho = 32$ for varying $E_{\text{given}}(G)$.

	HEFT		MSLECC		EECC	
$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
6140.20	12280.4	3137	6140.20	5868.63	6139.98	3917.47
7368.24	12280.4	3137	7368.24	5453.65	7368.04	3981.73
8596.28	12280.4	3137	8596.28	5003.22	8594.20	3668.03
9824.32	12280.4	3137	9824.32	4624.76	9805.49	3501.16
11052.36	12280.4	3137	11052.36	4266.21	10856.98	3544.09

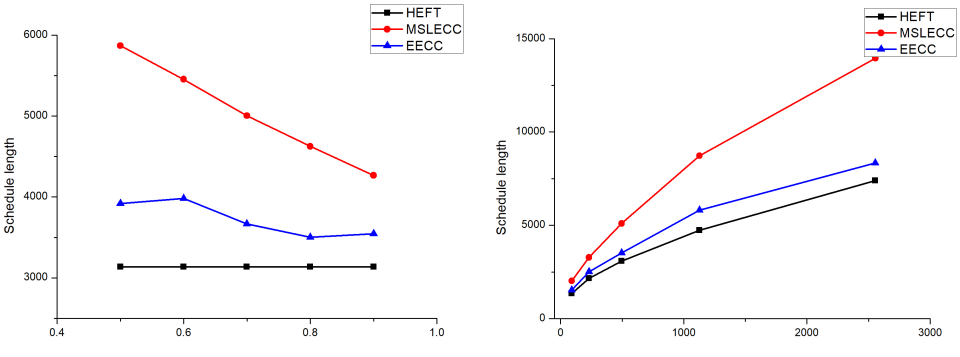
5.2. *Gaussian elimination application*

Similarly, in Gaussian elimination (GE) application, we define ρ as the dimension of the application, and the total number of tasks can be calculated by $|N| = \frac{(\rho^2+\rho-2)}{2}$. Since the FFT has higher parallelism than GE, it can be seen that the FFT can produce shorter scheduling length than GE (Fig. 5).

Experiment 3. This experiment compares the actual energy consumption and the final schedule length values of GE application for varying energy consumption constraints. We limit the matrix dimension as $\rho = 32$ (i.e., $|N| = 527$), and the energy consumption constraints is changed from $E_{\text{HEFT}}(G) \times 0.5$ to $E_{\text{HEFT}}(G) \times 0.9$.

Table 7 and Fig. 6(a) show the details of the final schedule lengths and energy consumption values of GE application with $\rho = 32$ for varying $E_{\text{given}}(G)$ by using all the algorithms. Compared to MSLECC, the EECC's scheduling length is reduced by 16.9% to 33.2%. Similar to Experiment 1, the results still show that EECC performs better than MSLECC.

Experiment 4. This experiment compares the actual energy consumption and the final schedule length of the GE applications for varying number of tasks. The matrix dimension set is $\{13, 21, 31, 47, 71\}$, the corresponding number of tasks is {90,230,495,1127,2555}. $E_{\text{given}}(G)$ is set to $E_{\text{HEFT}}(G) \times 0.5$.



(a) Varying $E_{\text{given}}(G)$ (b) Varying number of tasks

Fig. 6. Final schedule length of *GE* application.

Table 8. Results of GE applications for varying number of tasks.

			HEFT		MSLECC		EECC	
ρ	$\|N\|$	$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
13	90	1057.03	2114.06	1340	1057.03	2021.97	1055.08	1528.11
21	230	2338.99	4677.98	2160	2338.99	3278.90	2338.97	2510.46
31	495	5610.79	11221.58	3086	5610.79	5098.89	5610.62	3517.95
47	1127	14601.34	29202.67	4738	14601.34	8723.79	14597.87	5802.84
71	2555	34310.89	68621.77	7396	34310.89	13954.94	34310.73	8339.82

Table 8 and Fig. 6(b) show the results of Gaussian elimination applications for different number of tasks by using the three algorithms. Among these three algorithms, the MSLECC and EECC algorithms can meet the energy consumption constraints in any case, and as the scale increases, the EECC algorithm still has a better effect on the scheduling length than the MSLECC. The performance basically increased from 23.4% to 40.2%.

Through experiments on two real-world applications, fast Fourier transforms and Gaussian elimination, the results show that EECC algorithms can apply to different scale and parallel applications.

5.3. *Randomly generated parallel application*

For randomly generated graphs, we typically use a random DAG generator to randomly generate parallel applications. In this study, a parallel application is randomly generated based on the following parameters: communication to computation ratio (CCR) is 1, average computation time is 50 h, and shape parameter is 1. The value of the heterogeneity factor is in the range (0,1], where 0 and 1 indicate the lowest and highest heterogeneity factors, respectively.

Experiment 5. We conducted this experiment to compare the actual energy consumption and the final schedule length of low-heterogeneity (with the heterogeneity factor 0.1) and high-heterogeneity (with the heterogeneity factor 1.0) randomly generated parallel applications for varying energy consumption constraints, respectively. The number of tasks is set to $|N| = 511$, and the energy consumption constraints is changed from $E_{\text{HEFT}}(G) \times 0.5$ to $E_{\text{HEFT}}(G) \times 0.9$.

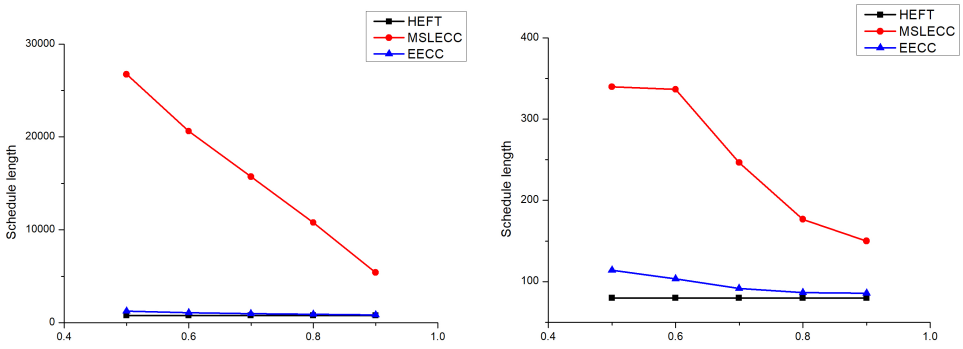
Tables 9 and 10, respectively, show the details of the final schedule lengths and energy consumption values of low-heterogeneity and high-heterogeneity randomly generated parallel applications with different energy consumption constraints by using three different algorithms. Because the objective computing platform is composed of heterogeneous processors, the heterogeneity may also affect the performance of the application. Figure 7 intuitively shows that with the increase of the heterogeneity factor, the performance of each scheduling algorithm has been improved to varying degrees. Similar to Experiment 1, these results show that EECC still performs better than MSLECC.

Table 9. Results of low-heterogeneity randomly generated parallel application with $|N| = 511$ for varying $E_{\text{given}}(G)$.

	HEFT		MSLECC		EECC	
$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
12971.17	25942.34	775	12971.16	26742.22	12970.99	1227.63
15565.40	25942.34	775	15565.40	20619.04	15564.87	1083.73
18159.64	25942.34	775	18159.63	15722.35	18159.42	978.24
20753.87	25942.34	775	20753.87	10778.17	20753.25	900.10
23348.11	25942.34	775	23348.10	5397.70	23342.03	827

Table 10. Results of high-heterogeneity randomly generated parallel application with $|N| = 511$ for varying $E_{\text{given}}(G)$.

	HEFT		MSLECC		EECC	
$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
1297.47	2594.93	80	1297.47	339.64	1296.77	114.06
1556.96	2594.93	80	1556.96	336.46	1555.23	103.53
1816.45	2594.93	80	1816.45	246.60	1793.36	91.54
2075.94	2594.93	80	2075.94	176.74	2012.64	86.47
2335.44	2594.93	80	2335.44	149.91	2215.98	85.62

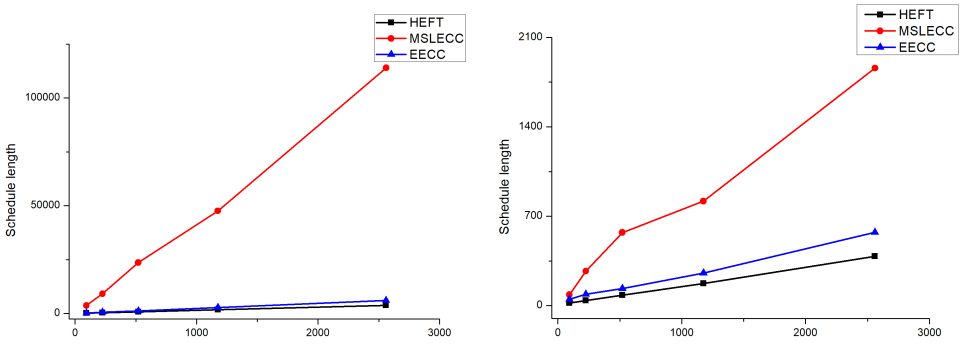


(a) Low-heterogeneity application (b) High-heterogeneity application

Fig. 7. Final schedule length for varying $E_{\text{given}}(G)$.

Experiment 6. We carried out this experiment to compare the actual energy consumption and the final schedule length of low-heterogeneity (with the heterogeneity factor 0.1) and high-heterogeneity (with the heterogeneity factor 1.0) randomly generated parallel applications for different number of tasks, respectively (Fig. 8). The number of tasks is changed from 93 to 2560. $E_{\text{given}}(G)$ is set to $E_{\text{HEFT}}(G) \times 0.5$.

Tables 11 and 12 show the results of low-heterogeneity and high-heterogeneity randomly generated parallel applications with different number of tasks by using three



(a) Low-heterogeneity application (b) High-heterogeneity application

Fig. 8. Final schedule length for varying number of tasks.

Table 11. Results of low-heterogeneity randomly generated parallel application for varying number of tasks.

		HEFT		MSLECC		EECC	
$\|N\|$	$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
93	2299.47	4598.93	142	2299.47	3736.10	2299.16	262.54
225	5609.54	11219.07	369	5609.53	9177.00	5609.41	576.43
520	13078.86	26157.72	793	13078.86	23636.42	13078.71	1270.51
1175	29833.63	59667.26	1775	29833.63	47573.33	29833.31	2812.26
2560	65194.25	130388.49	3846	65194.24	113973.50	65193.81	6088.99

Table 12. Results of high-heterogeneity randomly generated parallel application for varying number of tasks.

		HEFT		MSLECC		EECC	
$\|N\|$	$E_{\text{given}}(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
93	217.02	434.03	21	217.02	86	216.99	51.58
225	569.12	1138.23	40	569.12	270	569.08	90.91
520	1305.55	2611.10	83	1305.55	573	1304.03	135.24
1175	2765.15	5530.29	173	2765.15	818	2765.10	255.27
2560	6242.70	12485.39	386	6242.70	1859.52	6242.67	574.59

different algorithms. The results still show that the proposed EECC algorithm can not only be applied to small-scale applications, but also can be applied to large-scale applications, and has extensively prove the performance advantages of the proposed algorithm. Among them, the main difference between low-heterogeneity and high-heterogeneity applications is that low-heterogeneity application generates about 10 times more energy consumption and schedule length than high-heterogeneity application. In other words, application with a high degree of heterogeneity may have higher energy savings and the potential to reduce the schedule length.

6. Conclusion

This paper has presented an enhanced algorithm called EECC designed to minimize the schedule length of energy constrained parallel applications in heterogeneous distributed systems. First, the mathematical proof and experiments verify that the proposed algorithm can always satisfy the energy consumption constraints. Second, the algorithm can efficiently reduce the schedule length of the application with low time complexity. The EECC algorithm effectively improves the partial energy-aware design of parallel applications in heterogeneous distributed systems.

Acknowledgment

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0202901, Grant 2017YFB0202905, the National Natural Science Foundation of China under Grant 61702172, Grant 61672217, the Natural Science Foundation of Hunan Province under Grant 2018JJ3076, the Open Research Project of the State Key Laboratory of Synthetical Automation for Process Industries (SAPI), Northeastern University, China under Grant PAL-N201803, and the Fundamental Research Funds for the Central Universities.

References

1. A. Jaiantilal, Y. Jiang and S. Mishra, Modeling cpu energy consumption for energy efficient scheduling, in *The Workshop on Green Computing* (2010), pp. 10–15.
2. Y.-C. Hung, S.-H. Shieh and C.-K. Tung, A survey of low-voltage low-power techniques and challenges for cmos digital circuits, *J. Circuits Syst. Comput.* **20**(1) (2011) 89–105.
3. M. E. Salehi, M. Samadi, M. Najibi, A. Afzali-Kusha, M. Pedram and S. M. Fakhraie, Dynamic voltage and frequency scheduling for embedded processors considering power/performance tradeoffs, *IEEE Trans. Very Large Scale Integr. Syst.* **19**(10) (2011) 1931–1935.
4. C. Kahraman and O. Engin, Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach, *Appl. Soft Comput.* **10**(4) (2010) 1293–1300.
5. B. Keshanchi and N. J. Navimipour, Priority-based task scheduling in the cloud systems using a memetic algorithm, *J. Circuits Syst. Comput.* **25**(10) (2016), p. 1650119.
6. J. D. Ullman, Np-complete scheduling problems, *J. Comput. Syst. Sci.* **10**(3) (1975) 384–393.
7. T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao and D. Hensgen, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, *J. Parallel Distributed Comput.* **61**(6) (2001) 810–837.
8. G. Xie, G. Zeng, Y. Liu, J. Zhou, R. Li and K. Li, Fast functional safety verification for distributed automotive applications during early design phase, *IEEE Trans. Industr. Electron.* **65** (2018) 4378–4391.
9. G. Xie, G. Zeng, Z. Li, R. Li and K. Li, Adaptive dynamic scheduling on multifunctional mixed-criticality automotive cyber-physical systems, *IEEE Trans. Vehicular Technol.* **66**(8) (2017) 6676–6692.

10. G. Xie, H. Peng, Z. Li, J. Song, Y. Xie, R. Li and K. Li, Reliability enhancement towards functional safety goal assurance in energy-aware automotive cyber-physical systems, *IEEE Trans. Industr. Inform.* **PP**(2018) 1–14.
11. J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan and Y. Ma, Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms, *J. Syst. Softw.* **133** (2017) 1–16.
12. J. Zhou and T. Wei, Stochastic thermal-aware real-time task scheduling with considerations of soft errors, *J. Syst. Softw.* **102** (2015) 123–133.
13. J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu and Y. Ma, Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **35**(8) (2016) 1269–1282.
14. T. Wei, J. Zhou, K. Cao, P. Cong, M. Chen, G. Zhang, X. S. Hu and J. Yan, Cost-constrained qos optimization for approximate computation real-time tasks in heterogeneous mpsocs, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **PP**(99) (2017) 1–1.
15. X. Xiao, G. Xie, R. Li and K. Li, Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems, *Proc. 14th IEEE Int. Symp. Parallel and Distributed Processing with Applications* (IEEE, 2016), pp. 1471–1476.
16. X. Zhong and C. Z. Xu, Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee, *IEEE Trans. Comput.* **56**(3) (2007) 358–372.
17. K. H. Kim, R. Buyya and J. Kim, Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters, in *IEEE Int. Symp. CLUSTER Computing and the Grid* (2007), pp. 541–548.
18. G. Xie, G. Zeng, J. Jiang, C. Fan, R. Li and K. Li, Energy management for multiple real-time workflows on cyber–physical cloud systems, *Future Generation Computer Systems* (May, 2017).
19. Y. Chen, G. Xie and R. Li, Reducing energy consumption with cost budget using available budget preassignment in heterogeneous cloud computing systems, *IEEE Access* **6** (2018) 20572–20583.
20. G. Xie, X. Xiao, R. Li and K. Li, Schedule length minimization of parallel applications with energy consumption constraints using heuristics on heterogeneous distributed systems, *Concurr. Comput. Practice Exp.* **29** (2017), p. e4024.
21. G. Xie, G. Zeng, X. Xiao, R. Li and K. Li, Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems, *IEEE Trans. Parallel Distributed Syst.* **28**(12) (2017) 3426–3442.
22. H. Topcuoglu, S. Hariri and M. Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, *IEEE Trans. Parallel Distributed Syst.* **13**(3) (2002) 260–274.
23. 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* **22**(8) (2011) 1374–1381.
24. Q. Huang, S. Su, J. Li, P. Xu, K. Shuang and X. Huang, Enhanced energy-efficient scheduling for parallel applications in cloud, in *IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing* (2012), pp. 781–786.
25. G. Xie, J. Jiang, Y. Liu, R. Li and K. Li, Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems, *IEEE Trans. Industr. Informat.* **13** (2017) 1068–1078.

26. G. Xie, G. Zeng, R. Li and K. Li, Quantitative fault-tolerance for reliable workflows on heterogeneous iaas clouds, *IEEE Trans. Cloud Comput.* (2017), pp. 1–14, doi: 10.1109/TCC.2017.2780098.
27. D. Zhu, R. Melhem and D. Mosse, The effects of energy management on reliability in real-time embedded systems, in *IEEE/ACM Int. Conf. Computer Aided Design* (2004), pp. 35–40.
28. B. Zhao, H. Aydin and D. Zhu, On maximizing reliability of real-time embedded applications under hard energy constraint, *IEEE Trans. Industr. Inform.* **6**(3) (2010) 316–328.
29. B. Zhao, H. Aydin and D. Zhu, Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints, *ACM Trans. Design Autom. Electron. Syst.* **18**(2) (2013) 1–21.