Journal of Systems Architecture 122 (2022) 102329

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

HEA-PAS: A hybrid energy allocation strategy for parallel applications scheduling on heterogeneous computing systems

Jiwu Peng [a,b], Kenli Li [a,b,*], Jianguo Chen [a,b], Keqin Li [a,c]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[b] National Supercomputing Center in Changsha, Hunan 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Dynamical pre-allocate energy
Energy allocation
Heterogeneous computing systems
Precedence-constrained tasks
Static pre-allocate energy

ABSTRACT

Heterogeneous Computing Systems (HCS) have received widespread attention due to their powerful computing power, low cost, and high scalability. In HCS ranging from small-embedded devices to large data centers, energy consumption is one of the crucial design constraints. Meanwhile, the schedule length (response time) of parallel applications directly affects their Quality of Service (QoS) experience. In this study, we address the problem of minimizing the schedule length of energy-constrained parallel applications (MSLEC) on heterogeneous computing systems. Firstly, we define the concept of task energy demand rate and energy allocation factor to reasonably allocate the allocatable energy. Secondly, We propose a two-stage hybrid energy allocation (HEA) strategy and divide the allocatable energy into two parts according to the energy allocation factor, namely static pre-allocate energy (SAE) and dynamic pre-allocate energy (DAE). In the first stage, we pre-allocate SAE for each task based on the minimum energy demand and energy demand rate before task scheduling. In the second stage, we dynamically allocate DAE to each task during the operation of the scheduling algorithm. Thirdly, We conduct a rigorous mathematical proof of the feasibility of the proposed strategy. Finally, according to the proposed strategy, we design a novel HEA-based parallel application scheduling (HEA-PAS) algorithm, which aims to solve the MSLEC problem. Experiments on real-world and randomly generated parallel applications show that the proposed HEA-PAS algorithm outperforms the state-of-the-art methods in terms of effectiveness.

1. Introduction

1.1. Background

Heterogeneous Computing Systems (HCS) are widely used in scientific computing, industrial control, and mobile computing, etc. due to their strong computing power, low cost, and high scalability [1]. In HCS energy consumption is a core design indicator constraints [2–5]. To save energy and protect the environment, various management technologies have been established to maximize energy efficiency. Dynamic Voltage and Frequency Scaling (DVFS) is a well-known and effective energy optimization technique and a common technique in heterogeneous computing systems [6,7]. The current mainstream commercial processor series, such as Intel Atom, AMD Fusion, and ARM Cortex A9 are all support DVFS technology [8]. The basic idea is to dynamically adjust the operating voltage or frequency of the system components without affecting the operating efficiency of the system to optimize energy consumption. However, the performance and energy consumption of a computer system are always inseparable. Once the performance is improved, the corresponding energy consumption will increase to a certain extent. In practical applications, how to reach a balance between energy consumption and performance to meet requirement of green computing is a hot in the industrial and academic fields [9,10].

In a heterogeneous computing system, energy consumption and application schedule length (also termed as response time, complete time, makespan) are two mutually dependent optimization goals [11,12]. Energy consumption optimization mainly includes two branches: one is to minimize energy consumption under a given application schedule length constraint, and the other is to minimize the application schedule length under a given energy consumption constraint. In this study, we adopt the second research model to design an algorithm for minimizing the schedule length under limited energy constraints. Meanwhile, parallel task scheduling is critical to the performance of computing systems and more complicated in heterogeneous computing environments. According to the relationship between tasks, task

* Corresponding author at: College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China.
E-mail addresses: jiwu_peng@hnu.edu.cn (J. Peng), lkl@hnu.edu.cn (K. Li), jianguochen@hnu.edu.cn (J. Chen), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.sysarc.2021.102329
Received 16 June 2021; Received in revised form 8 October 2021; Accepted 30 October 2021
Available online 13 November 2021

scheduling can be divided into independent and non-independent task scheduling. Dependent tasks are a set of tasks with priority constraints, usually are modeled by Directed Acyclic Graph (DAG), and have been widely used in the current industry, such as Apache Spark [13] and Tensor Flow [14]. The task scheduling problem that satisfies the task priority constraint relationship is essentially a combinatorial optimization problem. The optimal solution of this type of problem is an NP-hard problem, that is, no optimal scheduling scheme can be found in polynomial time [15]. Therefore, heuristic algorithms and meta-heuristic algorithms are usually used to solve task scheduling problems in practical applications.

1.2. Motivation

Energy consumption and schedule length are one of the core indicators of parallel application scheduling in heterogeneous computing systems. In recent years, minimizing the schedule length in the energy-constrained parallel applications scheduling optimization has become a hot issue of research. The core of energy-constrained DAG-based parallel application scheduling lies in pre-allocating energy for each task. Based on the way that allocable energy is allocated to each task, we summarize the existing work into fully dynamic pre-allocation and fully static pre-allocation. In [16], a fully dynamic method is proposed, that is, the allocatable energy is pre-allocated to each task during the operation of the scheduling algorithm. However, this fully dynamic method result in a large amount of allocable energy being divided by high-priority tasks, while low-priority tasks pre-allocated too little energy and require more execution time to complete, so the application schedule length is not be optimistic. In [17], a static method of pre-allocating allocatable energy equally for each task was proposed. In [18], a static method of pre-allocating energy to each task according to energy level is proposed. However, the fully static approach does not take into account the fact that high-priority and critical tasks require more energy. Therefore, how to balance the energy consumption requirements of high-priority tasks and low-priority tasks has become a key issue. In this article, we propose a hybrid energy pre-allocation method that takes into account the energy requirements of high-priority and low-priority tasks, so as to minimize the schedule length of energy-constrained parallel applications.

1.3. Main contributions

In this study, we focus on the problem of minimizing the schedule length of parallel applications under energy constraints in HCS. We firstly introduce the concepts of energy allocation factor and energy demand rate, then we propose a novel two-stage hybrid energy allocation strategy task scheduling algorithm for energy-constrained parallel applications (HEA-PAS), which balances the different energy consumption requirements of high-priority tasks and low-priority tasks. The main contributions of this article can be summarized as follows:

- We define the concept of task energy demand rate and energy allocation factor to allocation the allocatable energy reasonably.
- We propose an efficient two-stage energy allocation strategy to balance the different energy consumption requirements of high-priority tasks and low-priority tasks, and prove the feasibility of the allocation strategy.
- We design a novel task scheduling algorithm (HEA-PAS) to schedule parallel applications, which aim to minimize schedule length while satisfying energy consumption constraints.
- We use real-world and randomly generated parallel applications to verify the effectiveness and scalability of the proposed algorithm. The results show that HEA-PAS can obtain minimum schedule length under different conditions compared with state-of-the-art algorithms.

The rest of this article is organized as follows: Section 2 reviews related work. Section 3 introduces the models of parallel applications and energy consumption. Section 4 introduces the main idea of the HEA strategy and HEA-PAS algorithm. Section 5 conducts the comparison experiments. Section 6 summarizes this work.

2. Related work

In recent years, task scheduling in heterogeneous computing systems has received extensive attention from researchers. In [19], Chen et al. studied the scheduling of data-dependent periodic tasks on heterogeneous multi-processor platforms. In [20,21] the schedulability of independent periodic tasks is analyzed. Energy consumption and schedule length are one of the core indicators of parallel application scheduling in heterogeneous computing systems. Heuristic methods are often used to solve this type of problem. For example, the work of [16–18] and others are based on heuristic algorithms. The meta-heuristic method is also one of the key methods to solve this kind of problems. In [22], Liu et al. studied the problem of maximizing reliability and minimizing scheduling length using the tabu search method. In [23], Kumar et al. used the discrete particle swarm method to minimize the power consumption and scheduling length of sequential tasks and parallel tasks. In [24], Yu et al. studied the optimization of makespan using the discrete intrusion weed algorithm on heterogeneous clusters, however, they did not consider energy consumption.

Various studies on energy consumption for distributed and parallel applications have been proposed in [25–27]. In [25], Zong et al. studied the energy aware scheduling problem in isomorphic systems by considering task replication and proposed energy-aware replication and performance-energy balanced replication algorithms to achieve the comprehensive consideration of system performance and energy. In [26], Huang et al. investigated energy consumption optimization under the constraint of application scheduling length in HCS, and proposed a energy-efficient scheduling algorithm. In [27], Xie et al. presented a global energy-saving scheduling algorithm, which supports DVFS by moving tasks to the processors with minimum dynamic energy consumption in HCS. Although above studies have taken into account the issue of energy saving, but it is not the same as our solution. Schedule Length optimization is another core issue that has been widely studied in the field of distributed and parallel applications on HCS [28–30]. In [28], Topcuoglu et al. proposed the well-known heterogeneous earliest completion time (HEFT) algorithm, which is used to schedule applications with task priority requirements on heterogeneous processors to achieve high performance goals. Cao et al. studied static heat-aware task allocation and scheduling methods to minimize schedule length in heterogeneous real-time MPSoC [29]. In [30], Zhou et al. presented an improved HEFT algorithm based on fuzzy dominance sorting to minimize the cost and schedule length of workflow in a cloud computing environment. In addition to the above work, most other studies only focus on reducing application energy consumption and do not care about the schedule length, or reducing application schedule length without caring about energy consumption.

Recently, solving the MSLEC problem on HCS has received a lot of attention [16–18]. In [16], Xiao et al. proposed an algorithm (MSLECC), which is based on DVFS technology to solve the problem of MSLEC parallel applications in HCS. This method regards the allocatable energy consumption as a whole, and then gradually shifts it to each task, giving priority to high-priority tasks, which is unfair to tasks with lower priority. We call this work as full dynamic pre-allocate method. In [17], Song et al. adopted a scheduling strategy that allocates available energy to each task evenly in advance is adopted, which we called full static pre-allocate method. This method balances the shortcomings of full dynamic allocation to a certain extent, but it cannot effectively solve the problem. In latest work [18], Quan et al. defined the concept of energy consumption level and pre-allocated available energy to each task according to the energy consumption level strategy. They considered the different energy requirements of each task to a certain extent. However, they did not consider the energy requirements of different priority tasks and critical tasks.

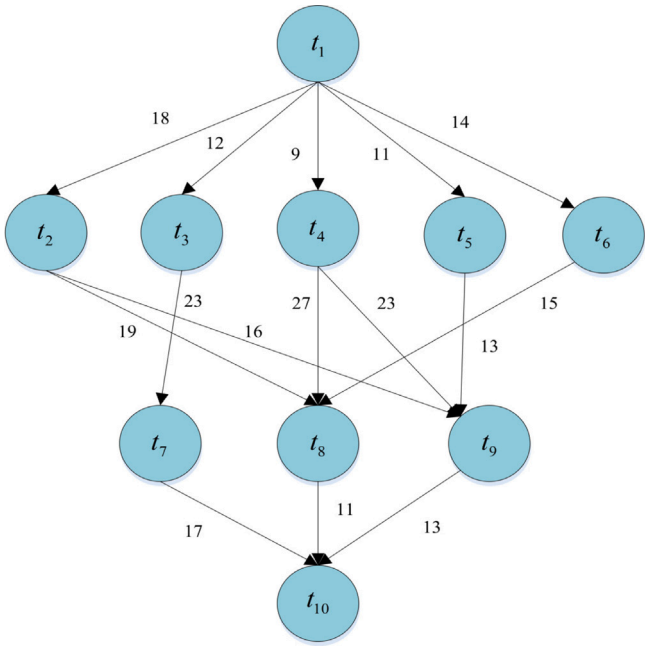


Fig. 1. Standard example of a DAG-based parallel application [18,28,31,32].

Table 1
The main parameters in this article.

Symbol	Meaning
P	The processor set of heterogeneous computing systems.
G	The DAG-based application model.
T	The task set of the application.
$\|T\|$	The number of task in the application.
M	The set of communication edges in application G.
$\mathcal{P}_{k,\text{ind}}$	The frequency-independent power in processor p_k.
$C_{k,ef}$	The effective switched capacitance in processor p_k.
m_k	The dynamic power exponent in processor p_k.
α	The energy allocation factor of the application G.

3. Models and preliminaries

The main parameters of this article are listed in Table 1. The main notations and their definitions of this article are given in Table 2.

3.1. Application model

Same as previous studies [18,28,31,32], the DAG application model is defined as $G = (T, E, C, W)$, where T represents the node set in application, E is the communication edge set, C is the communication time set, and W is computing matrix. $P = \{p_1, p_2, \ldots, p_{|P|}\}$ represents the processor set, where $|P|$ is the number of processors. Each node $t_i \in T$ represents a task, $c_{i,j} \in C$ is the communication time between tasks t_i and t_j, and $w_{i,k}$ is the execution time with maximum frequency of task t_i running on processor p_k. $pred(t_i)$ and $succ(t_i)$ represent the direct predecessor and successor set of task t_i, respectively. t_{entry} represents the entry task of the application and t_{exit} represents the exit task of the application.

3.2. Energy model

In this study, the power model we use following articles [18,27,33]. Specifically, when the frequency is f, the system power consumption is defined as:

$$\mathcal{P}(f) = \mathcal{P}_{\text{s}} + h\left(\mathcal{P}_{\text{ind}} + \mathcal{P}_d\right) = \mathcal{P}_{\text{s}} + h\left(\mathcal{P}_{\text{ind}} + C_{\text{ef}} f^m\right), \tag{1}$$

Table 2
Important notations in this article.

Symbol	Meaning
$EST(t_i, p_k)$	The earliest start time of task t_i execute on processor p_k.
$EFT(t_i, p_k)$	The earliest finish time of task t_i execute on processor p_k.
$AST(t_i)$	The actual execute time of task t_i.
$AFT(t_i)$	The actual finish time of task t_i.
$SL(G)$	The final schedule length of application G.
$E\left(t_i, p_k, f_{k,h}\right)$	The energy consumption of task t_i at frequency $f_{k,h}$.
$E(G)$	The total energy consumption of the application G.
$E_{\text{ae}}(G)$	The allocable energy of the application G.
$EDR(t_i)$	The energy demand rate of each task in the application.
$E_{\text{tae}}(t_i)$	The task allocation energy of each task in the application.
$E_{\text{pre}}(t_i)$	The pre-allocate energy consumption for task t_i
$E_{\text{cons}}(t_i)$	The energy consumption constraint of task t_i.
$E_{\text{cons}}(G)$	The energy consumption constraint of application G.

where $\mathcal{P}_s$ represents static power (same to [18], in this article we also do not consider it, because it is unmanageable), $\mathcal{P}_{ind}$ is frequency-independent power and $\mathcal{P}_d$ is frequency-dependent dynamic power, h represents the system state, C_{ef} is the effective capacitance, and m is the dynamic power exponent. The actual frequency f should be in the interval $[f_{\text{low}}, f_{\max}]$, where f_{low} is defined as $f_{\text{low}} = \max\left(f_{\min}, f_{\text{ee}}\right)$. The energy-efficient frequency denoted by f_{ee}, is computed as:

$$f_{ee} = \sqrt[m]{\frac{\mathcal{P}_{\text{ind}}}{(m-1)C_{\text{ef}}}}. \tag{2}$$

In a heterogeneous distributed system, we use $|P|$ to represent the number of processors, and the processors are heterogeneous, each processor has its own parameters. For a heterogeneous system, we can define the following sets:

$$\begin{cases} \mathcal{P}_{\text{ind}} = \left\{\mathcal{P}_{\text{ind},1}, \mathcal{P}_{\text{ind},2}, \ldots, \mathcal{P}_{\text{ind},|P|}\right\} \\ \mathcal{P}_d = \left\{\mathcal{P}_{d,1}, \mathcal{P}_{d,2}, \ldots, \mathcal{P}_{d,|P|}\right\} \\ C_{\text{ef}} = \left\{C_{\text{ef},1}, C_{\text{ef},2}, \ldots, C_{\text{ef},|P|}\right\} \\ m = \left\{m_1, m_2, \ldots, m_{|P|}\right\}. \\ f_{\text{low}} = \{f_{1,\text{low}}, f_{2,\text{low}}, \ldots, f_{|P|,\text{low}}\} \end{cases} \tag{3}$$

The actual effective frequency set is defined as:

$$f_{\text{ae}} = \left\{ \begin{array}{c} \left\{f_{\text{low},1}, f_{k,1}, \ldots, f_{\max 1,}\right\} \\ \left\{f_{\text{low},2}, f_{k,2}, \ldots, f_{\max,2}\right\} \\ \cdots, \\ \left\{f_{\text{low},|P|}, f_{k,|P|}, \ldots, f_{\max,|P|}\right\} \end{array} \right\}. \tag{4}$$

Hence, we can calculate the energy consumption of task t_i executed on the processor p_k with frequency $f_{k,h}$ in Eq. (5):

$$E\left(t_i, p_k, f_{k,h}\right) = \left(\mathcal{P}_{\text{ind}} + C_{\text{k,ef}} \times \left(f_{k,h}\right)^{m_k}\right) \times w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{5}$$

Therefore, the energy consumption of application is calculated in Eq. (6):

$$E(G) = \sum_{i=1}^{|T|} E\left(t_i, p_k, f_{k,h}\right). \tag{6}$$

The minimum and maximum energy consumption of task t_i are calculated by:

$$\begin{cases} E_{\min}\left(t_i\right) = \min\limits_{p_k \in U} E\left(t_i, p_k, f_{k,\text{low}}\right) \\ E_{\max}\left(t_i\right) = \max\limits_{p_k \in U} E\left(t_i, p_k, f_{k,\max}\right). \end{cases} \tag{7}$$

The minimum and maximum energy consumption of the application G are calculated by:

$$\begin{cases} E_{\min}(G) = \sum_{i=1}^{|T|} E_{\min}\left(t_i\right) \\ E_{\max}(G) = \sum_{i=1}^{|T|} E_{\max}\left(t_i\right). \end{cases} \tag{8}$$

As in [16–18], in this article we assume that the energy constraint $E_{cons}(G)$ is $E_{min}(G) \le E_{cons}(G) \le E_{max}(G)$.

3.3. Execution model

The important concepts involved in the task execution model in this study are as follows:

(1) Task Prioritizing (TP): We use the ranking value of the HEFT algorithm [28] as the task priority criterion. The specific value $\mathrm{rank}_u(t_i)$ is given by

$$\mathrm{rank}_u(t_i) = \overline{w_i} + \max_{t_j \in \mathrm{succ}(t_i)} \left\{ c_{i,j} + \mathrm{rank}_u(t_i) \right\}, \tag{9}$$

where, the $\overline{w_i}$ is $\overline{w_i} = \left(\sum_{k=1}^{|P|} w_{i,k} \right) / |P|$.

*(2) Earliest Start Time (*EST*):* The EST of task t_i executed on processor p_k with frequency $f_{k,h}$ is denoted as $EST(t_i, p_k, f_{k,h})$, which is computed by

$$\begin{cases} \mathrm{EST}\left(t_{\mathrm{entry}}, p_k, f_{k,h}\right) = 0 \\ \mathrm{EST}\left(t_i, p_k, f_{k,h}\right) = \max\limits_{t_j \in \mathrm{pred}(t_i)} \left\{ \mathrm{avail}[k], \max\left\{ \mathrm{AFT}\left(t_j\right) + c'_{j,i} \right\} \right\}. \end{cases} \tag{10}$$

*(3) Earliest Finish Time (*EFT*):* The EFT of task t_i executed on processor p_k with frequency $f_{k,h}$ is denoted as $\mathrm{EFT}(t_i, p_k, f_{k,h})$, which is computed by

$$\mathrm{EFT}\left(t_i, p_k, f_{k,h}\right) = \mathrm{EST}\left(t_i, p_k, f_{k,h}\right) + w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{11}$$

(4) Schedule Length (SL): The $SL(G)$ is the entire execution time of a application from the entry task to the exit task using a scheduling algorithm, and its computed by

$$\mathrm{SL}(G) = \max_{t_i \in \text{exit task}} \mathrm{AFT}(t_i). \tag{12}$$

3.4. Problem description

In this work, we aim to solve the problem of minimizing the schedule length of energy-constrained parallel applications on heterogeneous computing systems. This optimization problem is described mathematically as the following:

$$\begin{aligned} \text{Minimize}: \quad & \mathrm{SL}(G) = \max_{t_i \in \text{exit task}} \mathrm{AFT}(t_i) \\ \text{Subject to}: \quad & E(G) = \sum_{i=1}^{|T|} E\left(t_i, p_k, f_{k,h}\right) \le E_{\mathrm{cons}}(G) \\ & E_{\min}(G) \le E_{\mathrm{cons}}(G) \le E_{\max}(G). \end{aligned} \tag{13}$$

3.5. The state-of-the-art algorithms

In recent work [18], Quan et al. the concept of energy consumption level is defined, and the assignable energy consumption is pre-allocate for each task according to the energy consumption level strategy.

Definition 1. Allocatable Energy (AE). Allocatable energy is the energy remaining after subtracting the minimum energy required by the application for a given energy, as defined in Eq. (14):

$$\Delta E_{\mathrm{ae}}(G) = E_{\mathrm{cons}}(G) - E_{\min}(G). \tag{14}$$

The energy consumption level of tasks and applications are calculated by:

$$\begin{cases} E_{\mathrm{ave}}\left(t_i\right) = \frac{E_{\max}(t_i) + E_{\min}(t_i)}{2} \\ E_{\mathrm{ave}}\left(G\right) = \frac{E_{\max}(G) + E_{\min}(G)}{2} \end{cases}. \tag{15}$$

The weight given by the author is calculated by:

$$\mathrm{el}\left(t_i\right) = \frac{E_{\mathrm{ave}}\left(t_i\right)}{E_{\mathrm{ave}}(G)}. \tag{16}$$

The pre-allocate energy consumption for each task is calculated by:

$$E_{\mathrm{pre}}\left(t_i\right) = E_{\mathrm{ae}}(G) \times \mathrm{el}\left(t_i\right) + E_{\min}\left(t_i\right). \tag{17}$$

In [16], each task is pre-allocate the minimum energy consumption, which results in high-priority tasks that can share most of the total allocatable energy. Conversely, due to the allocatable energy is small, low-priority tasks must find low-energy processors, which reduces the chance of obtaining optimistic schedule length. In other words, the pre-allocation strategy in MSLECC is extreme. Although the latest work [17] and [18] have balanced the deficiencies of the [16] article strategy to a certain extent, so that each task has a relatively fair energy. However, they do not take into account the different energy consumption requirements of high-priority tasks and low-priority tasks.

4. Our solution

The main idea of our algorithm is according to the energy allocation factor to divide the allocatable energy into static pre-allocate energy based on the energy demand rate (to meet the energy demand of low-priority tasks) and dynamic pre-allocate energy which transferred with the scheduling algorithm (to meet the energy demand of high-priority tasks). By meeting the different energy requirements of high-priority and low-priority tasks, we aim to minimize the schedule length of the parallel application. Fig. 2 provides a general flow of the proposed HEA-PAS. The specific process steps are described as follows: (1) Submit and build parallel applications based on the DAG-model, and prioritize them. (2) Calculate the maximum and minimum energy of each task and application according to the processor information. (3) Calculate the energy that can be allocated by the application and calculate the energy demand rate of each task. (4) Set the initial energy distribution factor and do the initial static pre-allocation. (5) Adjust the energy allocation factor, and carry out static and dynamic pre-allocation. S represents the static pre-allocated energy, D represents the dynamic pre-allocated energy, and the size of the square represents the amount of pre-allocated energy. (6) Find the optimal processor and frequency combination for parallel application tasks. (7) Execute the application according to the optimal combination.

4.1. The energy pre-allocate strategy

We first introduce a few concepts, these concepts help understand our pre-allocate strategy.

Definition 2. Energy Demand Rate (EDR). The energy demand rate is defined as the minimum energy demand of each task divided by the minimum energy demand of the application.

$$\mathrm{EDR}(t_i) = \frac{E_{\min}\left(t_i\right)}{E_{\min}(G)}. \tag{18}$$

Definition 3. Energy Allocation Factor (EAF). The energy allocation factor α is defined for dividing the allocatable energy into the static pre-allocate energy (to meet the energy demand of low-priority tasks) and the dynamic pre-allocate energy (to meet the energy demand of high-priority tasks).

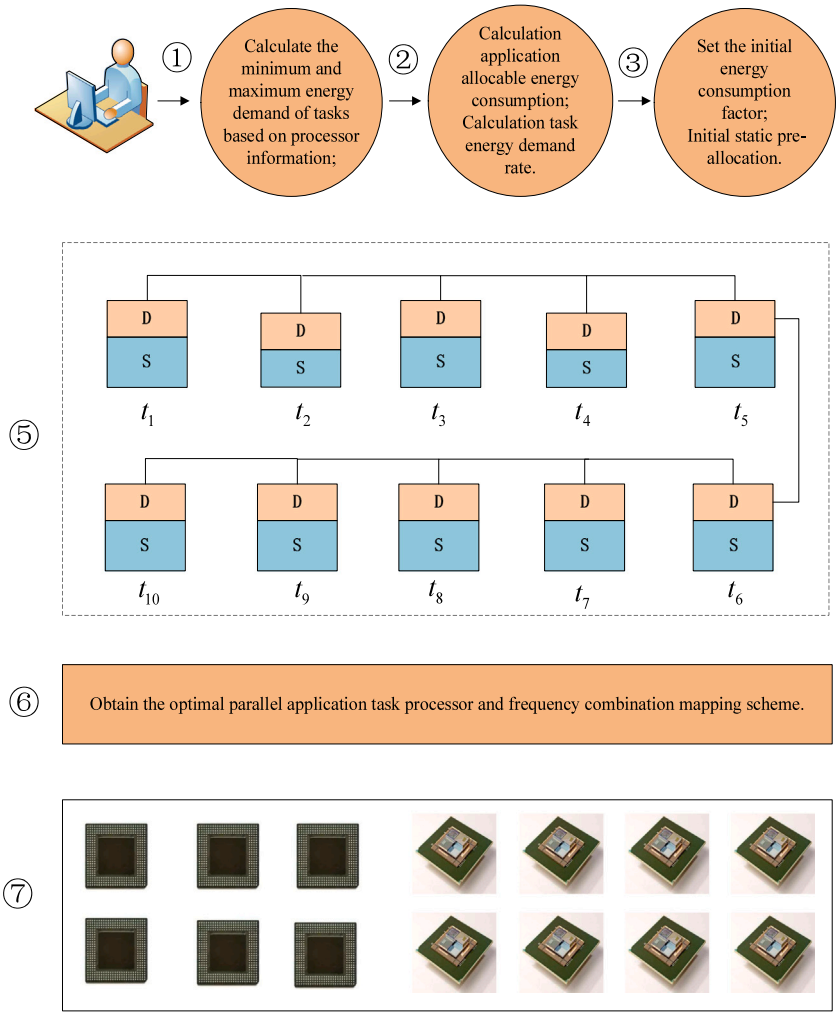


Fig. 2. General flow of HEA-PAS.

Definition 4. Static Pre-allocate Energy (SAE). SAE is defined as the energy constraint that is statically allocated to tasks in advance to meet the energy constraints of applications.

In this study, SAE can be calculated as the minimum energy consumption requirement of the task plus the allocatable energy consumption times the energy demand rate times the energy consumption distribution factor, as defined in Eq. (19):

$$E_{\text{sae}}\left(t_i\right) = E_{\min}\left(t_i\right) + \Delta E_{\text{ae}}(G) \times \text{EDR}(t_i) \times \alpha. \tag{19}$$

The energy allocated to the task does not need to exceed its maximum energy demand, so there is:

$$E_{\text{pre}}\left(t_i\right) = \min\left\{E_{\text{sae}}\left(t_i\right), E_{\max}\left(t_i\right)\right\}. \tag{20}$$

Definition 5. Application Assigned Energy (AAE). The energy consumed by the application refers to the energy consumed during the scheduling process:

$$\text{AAE}_{s(j)}(G) = \sum_{i=1}^{j-1} E\left(t_{s(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right). \tag{21}$$

Definition 6. Application Unassigned Energy (AUE). The energy consumed by the application refers to the unallocated energy consumption during the scheduling process:

$$\text{AUE}_{s(j)}(G) = \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right). \tag{22}$$

Definition 7. Residual Energy (RE). For task $t_{s(j)}$, its residual energy is defined in Eq. (23):

$$\begin{aligned} E_{\text{re}}(t_{s(j)}) = {} & E_{\text{cons}}(G) \\ & - \sum_{i=1}^{j} E\left(t_{s(j)}, p_{k(s(j))}, f_{k(s(j)),h(s(j))}\right) \\ & - \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right). \end{aligned} \tag{23}$$

We schedule tasks according to the priority queue of the upward sorting value. Assuming that the current task to be allocated is $t_{s(j)}$ then $\{t_{s(1)}, t_{s(2)}, \ldots, t_{s(j-1)}\}$ represents the assigned task set, and the unallocated task set consists of $\{t_{s(j+1)}, t_{s(j+2)}, \ldots, t_{s(|T|)}\}$. All tasks of the DAG application are unassigned at the beginning. To ensure that each task allocation meets the energy limit of the DAG application, and each unallocated task of our propose strategy provides pre-allocate according to Eqs. (20) and (21). Among them, the energy consumption of the DAG application is defined as:

$$\begin{aligned} E_{s(j)}(G) = {} & \sum_{i=1}^{j-1} E\left(t_{s(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right) \\ & + E\left(t_{s(j)}, p_k, f_{k,h}\right) + \sum_{i=1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right). \end{aligned} \tag{24}$$

According to the definition of the problem, $E_{s(j)}(G)$ must be less than or equal to $E_{\text{cons}}(G)$. Therefore, it is expressed as:

$$\begin{aligned}&\sum_{i=1}^{j-1} E\left(t_{s(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right)\\&\quad + E\left(t_{s(j)}, p_k, f_{k,h}\right) + \sum_{i=1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right)\\&\quad \le E_{\text{cons}}(G).\end{aligned} \tag{25}$$

Hence, the energy consumption of task $t_{s(j)}$ should have the following constraints:

$$\begin{aligned}&E\left(t_{\text{s}(j)}, p_k, f_{k,h}\right)\\&\le E_{\text{cons}}(G) - \sum_{i=1}^{j-1} E\left(t_{\text{s}(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right)\\&- \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right).\end{aligned} \tag{26}$$

Therefore, the energy consumption constraint given by task $t_{s(j)}$ is:

$$\begin{aligned}E_{\text{cons}}\left(t_{\text{s}(j)}\right) &= E_{\text{cons}}(G)\\&- \sum_{i=1}^{j-1} E\left(t_{\text{s}(j)}, p_{k(s(j))}, f_{k(s(j)),h(\text{s}(j))}\right)\\&- \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right).\end{aligned} \tag{27}$$

As long as each task meets its energy consumption constraint, that is, $E\left(t_{s(j)}, p_{k(s(j))}, f_{k(s(j)),h(s(j))}\right) \leqslant E_{\text{cons}}\left(t_{s(j)}\right)$, then the energy consumption constraint of the application will be satisfied. Hence, when processing task t_i, only the energy constraints of the task itself need to be considered, not the total energy consumption constraints of the application.

4.2. Feasibility of the proposed strategy

In this section, we give Theorem 1 to illustrate that our proposed strategy can satisfy the application energy consumption constraints when scheduling each task.

Theorem 1. *Given a DAG-based application G and energy constraints $E_{\min}(G) \le E_{\text{cons}}(G) \le E_{\max}(G)$, through the HEA strategy to pre-allocates the energy constraints of unscheduled tasks, each task $t_{s(j)}$ can always find a processor that satisfies:*

$$\begin{aligned}E_{s(j)}(G) &= \sum_{i=1}^{j-1} E\left(t_{s(j)}, p_{k(s(j))}, f_{k(s(j)),h(s(j))}\right)\\&+ E\left(t_{s(j)}, p_k, f_{k,h}\right) + \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right)\\&\le E_{\text{cons}}(G).\end{aligned} \tag{28}$$

Proof. Mathematical induction is used to prove Theorem 1. Firstly, for the entry task $t_{s(1)}$, all tasks are not allocated to the processor, and the application G should meet its energy consumption constraints:

$$\begin{aligned}E_{s(1)}(G) &= E\left(t_{s(1)}, p_k, f_{k,h}\right) + \sum_{i=2}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right)\\&\le E_{\text{cons}}(G).\end{aligned} \tag{29}$$

According to Eqs. (24), (25), (29), there is $E_{\text{pre}}(t_i) \le E_{\text{sae}}(t_i)$, so we can get:

$$\begin{aligned}&E\left(t_{s(1)}, p_k, f_{k,h}\right) + \sum_{i=2}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right)\\&\le E\left(t_{s(1)}, p_k, f_{k,h}\right) + \sum_{i=2}^{|T|} E_{\text{sae}}\left(t_{s(i)}\right),\end{aligned} \tag{30}$$

and have

$$\begin{aligned}&E\left(t_{\text{s}(1)}, p_k, f_{k,h}\right) + \sum_{i=2}^{|T|} E_{\text{sae}}\left(t_{\text{s}(i)}\right)\\&= E\left(t_{\text{s}(1)}, p_k, f_{k,h}\right) + \sum_{i=1}^{|T|} E_{\text{sae}}\left(t_{\text{s}(i)}\right)\\&- E_{\text{sae}}\left(t_{\text{s}(1)}\right)\\&\le E\left(t_{\text{s}(1)}, p_k, f_{k,h}\right) + \sum_{i=1}^{|T|} E_{\text{sae}}\left(t_{\text{s}(i)}\right)\\&- E_{sae}\left(t_{\text{s}(1)}\right) + E_{\text{re}}(t_{s(1)}).\end{aligned} \tag{31}$$

Obviously, according to Eq. (19), $E_{sae}\left(t_{\text{s}(1)}\right) + E_{\text{ex}}(t_{s(1)})$ is greater than or equal to $E_{\min}(t_{s(1)})$, so the processor with the energy consumption can at least be allocated to $t_{s(1)}$. Therefore, $t_{s(1)}$ can find an allocated processor and frequency to satisfy:

$$\begin{aligned}E_{\text{s}(1)}(G) &= E\left(t_{\text{s}(1)}, p_k, f_{k,h}\right) + \sum_{i=2}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right)\\&\le E_{\text{cons}}(G).\end{aligned} \tag{32}$$

Secondly, suppose that $t_{s(j)}$ can find a processor and the corresponding frequency to satisfy the constraint, and there is:

$$\begin{aligned}E_{\text{s}(j)}(G) &= \sum_{i=1}^{j-1} E\left(t_{\text{s}(i)}, p_{k(\text{s}(i))}, f_{k(\text{s}(i)),h(\text{s}(i))}\right)\\&+ E\left(t_{\text{s}(j)}, p_k, f_{k,h}\right) + \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right)\\&= \sum_{i=1}^{j} E\left(t_{s(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right)\\&+ \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right) \le E_{\text{cons}}\ (G).\end{aligned} \tag{33}$$

Hence,

$$\begin{aligned}&\sum_{i=1}^{j} E\left(t_{s(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right)\\&\le E_{\text{cons}}(G) - \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right).\end{aligned} \tag{34}$$

Therefore, for $t_{s(j+1)}$, the energy consumption of the parallel application G is:

$$\begin{aligned}E_{\text{s}(j+1)}(G) &= \sum_{i=1}^{j} E\left(t_{\text{s}(i)}, p_{k(\text{s}(i))}, f_{k(\text{s}(i)),h(\text{s}(i))}\right)\\&+ E\left(t_{\text{s}(j+1)}, p_k, f_{k,h}\right)\\&+ \sum_{i=j+2}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right).\end{aligned} \tag{35}$$

Then, substitute Eq. (34) into Eq. (35) to get

$$\begin{aligned}E_{\text{s}(j+1)}(G) &= \sum_{i=1}^{j} E\left(t_{\text{s}(i)}, p_{k(\text{s}(i))}, f_{k(\text{s}(i)),h(\text{s}(i))}\right)\\&+ E\left(t_{\text{s}(j+1)}, p_k, f_{k,h}\right) + \sum_{i=j+2}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right)\\&\le E_{\text{cons}}(G) - \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right)\\&+ E\left(t_{\text{s}(j+1)}, p_k, f_{k,h}\right) + \sum_{i=j+2}^{|T|} E_{\text{pre}}\left(t_{\text{s}(i)}\right)\\&\le E_{\text{cons}}(G) - E_{\text{pre}}\left(t_{\text{s}(j+1)}\right)\\&+ E\left(t_{\text{s}(j+1)}, p_k, f_{k,h}\right).\end{aligned} \tag{36}$$

Since the value of $E_{\text{pre}}(t_{(s)(j+1)})$ is greater than or equal to $E_{\min}(t_{s(j+1)})$, we can get a similar result when $j = 1$, that is $E_{s(j+1)}(G) \le E_{\text{cons}}(G)$.

This means that $t_{s(j+1)}$ can also find a specified processor to meet the energy constraints.

When all tasks can find a single allocated processor to meet the energy consumption constraint, the correctness of Theorem 1 proved. □

4.3. The proposed HEA-PAS algorithm

When assigning task $t_{s(j)}$, according to Theorem 1, the energy consumption constraint of this task can be given as:

$$\begin{aligned} & E_{\text{cons}}\left(t_{s(j)}, p_k, f_{k,h}\right) \\ & = E_{\text{cons}}(G) - \sum_{i=1}^{j-1} E\left(t_{s(i)}, p_{k(s(i))}, f_{k(s(i)),h(s(i))}\right) \\ & - \sum_{i=j+1}^{|T|} E_{\text{pre}}\left(t_{s(i)}\right). \end{aligned} \tag{37}$$

Therefore, after determining the given energy for each task, the hybrid energy allocation algorithm (HEA-PAS) proposed in Algorithm 1. In HEA-PAS, for each task traverses each processor to find the frequency with the EFT under a given energy consumption constraint. Adjust the energy allocation factor, repeat the above steps, and find the minimum schedule length of the application. The detailed description of each stage is as follows:

(1) Prioritization of tasks. In Line 1, HEA-PAS sort the tasks in parallel application by descending order of $rank_u(t_i)$ as S_{dsort}.

(2) Calculation of the minimum and maximum energy consumption of tasks and application. In Lines 2–5, HEA-PAS calculate minimum and maximum energy consumption of each task and the application by using Eqs. (7) and (8).

(3) Static energy pre-allocation (first stage). In Line 6–9, HEA-PAS calculate energy demand rate of each task and pre-allocate the SAE for each task, respectively (to meeting the energy requirements of low-priority tasks).

(4) Dynamic energy pre-allocation (second stage). In Lines 13–15, HEA-PAS calculate application assigned energy and application unassigned energy of scheduling task sequence $t_{s(j)}$ and pre-allocate the DAE for each task (to meeting the energy requirements of high-priority tasks).

(5) Satisfaction of energy constraints. In Lines 16–29, HEA-PAS traverse each processor and frequency to satisfy the energy constraint transferred to each task from the application energy constraint. Lines 19–21 skip the frequencies that are greater than the energy constraints of each task.

(6) Minimization of schedule length. In Lines 22–28, HEA-PAS minimize the schedule length of the application through minimize the earliest finish time of each task.

(7) Record of the optimal energy allocation factor. In Lines 34–38, HEA-PAS record the minimum scheduling length, energy consumption and the most effective energy allocation factor.

Complexity analysis. For traversing energy allocation factor consumes $O(|A|)$ time, where $|A|$ represents the number of energy allocation factor. For each task, selecting the processor and frequency with the EFT has complexity $O(|T|^2 \times |P| \times |F|)$, and $|F|$ is the number of discrete frequencies from $[f_{k,\text{low}}, f_{k,\text{max}}]$. Thus, the total time complexity is $O(|A| \times |T|^2 \times |P| \times |F|)$. Through a large number of experiments, we found that the optimal energy allocation factor is between 0.7 and 1. Therefore, we set A_s to 0.7, and the increase rate is 0.01, at this time $O(|A|) = 30$. Calculating the earliest start time only needs to traverse the predecessor nodes of the task, generally speaking, the value of $|T|$ is relatively small. Generally, HEA-PAS has higher time complexity for HEFT and other heuristic algorithms, but much lower complexity for media heuristic algorithms. For the same heterogeneous platform and the same application, only need to run once to find and fix the optimal energy distribution factor, and the complexity at this time is $O(|T|^2 \times |P| \times |F|)$, which is the same as HEFT.

Algorithm 1 The HEA-PAS Algorithm.

Require: $G = (T, E, C, W), P, E_{\text{cons}}(G)$.
Ensure: $\text{SL}(G)$, $E(G)$, $\alpha(G)$.
1: Sort tasks in the application by descending order of $rank_u(t_i)$ as S_{dsort};
2: **for** $(\forall i, t_i \in T)$ **do**
3: Calculate $E_{\text{min}}(t_i)$, $E_{\text{max}}(t_i)$ using Eq. (7);
4: **end for**
5: Calculate $E_{\text{min}}(G)$, $E_{\text{max}}(G)$ using Eq. (8);
6: **for** $(\forall i, t_i \in T)$ **do**
7: Calculate $\text{EDR}(t_i)$ using Eq. (18);
8: Calculate $E_{\text{pre}}(t_i)$ using Eqs. (19) and (20);
9: **end for**
10: **for** $(\forall j, \alpha(j) \in [A_s, 1]$ **do**
11: **while** tasks in S_{dsort} **do**
12: $t_i \leftarrow S_{dsort}$.out();
13: Calculate $\text{AAE}_{s(i)}(G)$ using Eq. (21);
14: Calculate $\text{AUE}_{s(i)}(G)$ using Eq. (22);
15: Calculate $E_{\text{cons}}(t_i)$ using Eq. (27);
16: **for** $(\forall k, p_k \in P)$ **do**
17: **for** $(f_{k,h} \in [f_{k,\text{low}}, f_{k,\text{max}}]$ **do**
18: Calculate $E(t_i, p_k, f_{k,h})$ using Eq. (5);
19: **if** $E(t_i, p_k, f_{k,h}) > E_{\text{cons}}(t_i)$ **then**
20: Continue;
21: **end if**
22: **if** $(\text{EFT}(t_i, p_k, f_{k,h}) < \text{AFT}(t_i))$ **then**
23: $E(t_i) \leftarrow E(t_i, p_k, f_{k,h})$;
24: $\text{AST}(t_i) \leftarrow \text{EST}(t_i, p_k, f_{k,h})$;
25: $\text{AFT}(t_i) \leftarrow \text{EFT}(t_i, p_k, f_{k,h})$;
26: $P(t_i) \leftarrow p_k$; //Record the actual allocation processor;
27: $F(t_i) \leftarrow f_{k,h}$; //Record the actual allocation frequency;
28: **end if**
29: **end for**
30: **end for**
31: Compute the $E_{\alpha(j)}(G)$ using Eq.(5);
32: Compute the $\text{SL}_{\alpha(j)}(G)$ using Eq.(12);
33: **end while**
34: **if** $\text{SL}_{\alpha(j)}(G) < \text{SL}(G)$ **then**
35: $E(G) \leftarrow E_{\alpha(j)}(G)$;
36: $\text{SL}(G) \leftarrow \text{SL}_{\alpha(j)}(G)$;
37: $\alpha(G) \leftarrow \alpha(j)$; //Record real energy allocation factor;
38: **end if**
39: **end for**
40: **return** $\text{SL}(G)$, $E(G)$, $\alpha(G)$

4.4. Example of the HEA-PAS algorithm

As in [16–18], we use Fig. 1 as an example of motivation. Table 3 lists the parameters of all tasks. Table 4 lists all processors parameters, such as dynamic power independent of frequency $P_{k,\text{ind}}$, effective switched capacitance $C_{k,e}$ and dynamic power index m_k. The maximum frequency $f_{k,\text{max}}$ of each processor is 1.0, and its frequency accuracy is 0.01. The minimum energy efficiency frequency $f_{k,\text{ee}}$ is derived from the Eq. (2). Therefore, according to the Eqs. (13) and (14), $E_{\text{min}}(G) = 20.31$ and $E_{\text{max}}(G) = 161.99$. We set the energy constraint of application G to $E_{\text{cons}}(G) \leq E_{\text{max}}(G)$, and we set $E_{\text{cons}}(G)$ to $E_{\text{HEFT}}(G) \times 0.5$. Then, Table 5 shows the task allocation of the parallel application in Fig. 1. The actual energy consumption of the application is 79.04, which is less than $E_{\text{cons}}(G)$. It can be seen from the motivation example that the schedule length of our algorithm is 80, which is the same as the HEFT algorithm, but the energy consumption of HEFT is 103.49, which exceeds the cons energy consumption $E_{\text{cons}}(G)$. Correspondingly, the schedule length of MSLECC [16] is 129.37, ESECC [17] is 84.03, and ISAECC [18] is 86.28. It can be seen from the example that our algorithm has a shorter schedule length than the previous work. For the sake of intuition, Fig. 3 describes the scheduling Gantt chart. The above example shows that our method can balance the different energy consumption requirements of high-priority tasks and low-priority tasks.

Table 3
Parameters of example tasks.

Task	p_1	p_2	p_3	$rank_u(t_i)$
t_1	14	16	9	108.000
t_3	11	13	19	80.000
t_4	13	8	17	80.000
t_2	13	19	18	77.000
t_5	12	13	10	69.000
t_6	13	16	9	63.333
t_9	18	12	20	44.333
t_7	7	15	11	42.667
t_8	5	11	14	35.667
t_{10}	21	7	16	14.667

Table 4
Parameters of example processors.

p_k	$\mathcal{P}_{k,\text{ind}}$	$C_{k,ef}$	m_k	$f_{k,\text{low}}$	$f_{k,\max}$
p_1	0.03	0.8	2.9	0.26	1.0
p_2	0.04	0.8	2.5	0.26	1.0
p_3	0.07	1.0	2.5	0.29	1.0

Table 5
The results of example application in Fig. 1 by using HEA-PAS algorithm.

Task	$E_{\text{cons}}(t_i)$	$p(t_i)$	$f(t_i)$	$AST(t_i)$	$AFT(t_i)$	$E(t_i)$	$E_{re}(t_i)$
t_1	18.2562	p_3	1.00	0.0	9.0	9.6300	8.6262
t_2	15.4061	p_3	0.84	9.0	30.4286	15.3577	0.0484
t_3	15.9253	p_1	1.00	21.0	32.0	9.1300	6.7953
t_4	13.6017	p_2	1.00	18.0	26.0	6.7200	6.8817
t_5	6.5295	p_3	0.67	30.4286	45.3539	6.5290	6.0E−4
t_6	7.0322	p_1	0.78	32.0	48.6667	6.9865	0.0457
t_7	5.4708	p_1	0.96	48.6667	55.9583	5.4008	0.0700
t_8	4.6883	p_1	1.00	55.9583	60.9583	4.1500	0.5383
t_9	9.3845	p_2	0.94	58.3539	71.1199	9.2597	0.1248
t_{10}	7.8312	p_2	1.00	72.9583	80.0	5.8800	1.9512

$E(G) = 79.0438 \le E_{\text{cons}}(G) = 80.995,\ SL(G) = SL_{\text{HEFT}}(G) = 80$

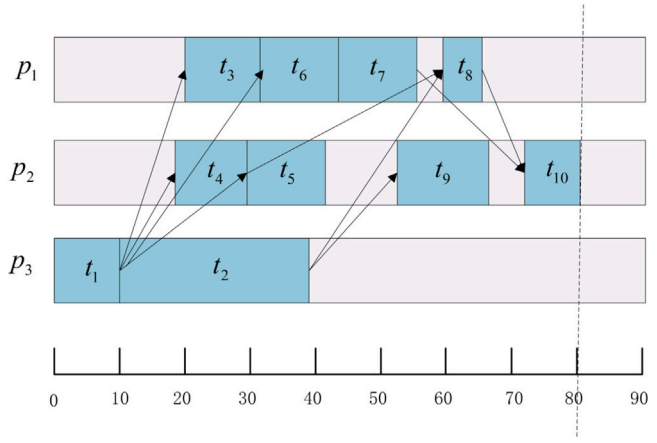


Fig. 3. The scheduling Gantt chart of application in Fig. 1 using HEA-PAS.

5. Experiments

To better illustrate the superiority of the propose algorithm, this section will do further comparative experiments. The algorithms participating in the comparison are HEFT [28], MSLECC [16], ESECC [17], ISAECC [18], HEA-PAS. The HEFT algorithm does not consider energy consumption constraints, and only minimizes the schedule length as an optimization objective. Since we use the energy consumption generated by the HEFT as the benchmark for energy consumption constraints (ie $E_{\text{HEFT}}(G)\times\lambda, 0 \le \lambda \le 1$), we use the results of the HEFT algorithm as the benchmark for comparison. The main idea of the MSLECC algorithm is to transfer the overall remaining energy consumption of the application

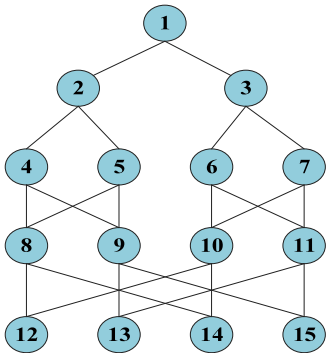


Fig. 4. Example of the FFT parallel application with $\rho = 4$.

to for each task, ESECC adopts a strategy of evenly distributing the remaining energy consumption to each task, and ISAECC adopts a weighting strategy at the application level, it is one of the best performing algorithms in the current research on this problem. We mainly compare the actual energy consumed $E(G)$ (Unit: kWs) [34] and the schedule length SL(G) (Unit: s) (application response time/complete time/makespan) by each algorithm. We use two real-world parallel applications Fast Fourier Transform (FFT) and Gaussian Elimination (GE) [18,28,35], and randomly generated applications to verify the related algorithms.

5.1. Experimental metrics

In this study, we focus on the task scheduling design phase of parallel applications. We conduct the experiments on a custom simulated heterogeneous platform with 64 processors. Like CloudSim [36], our simulator also uses java as the programming language. The widely used task graph generator TaskGraphGenerator [37] is used as a component to generate random parallel applications The parameters of the processors in the experiment are the same as in [18]. The detailed processor and application parameter ranges are set as follows: $10\text{s} \le \text{w}_{i,k} \le 100\text{s}, 10\text{s} \le c_{i,j} \le 100\text{s}, 0.03 \le \mathcal{P}_{k,\text{ind}} \le 0.07, 0.8 \le C_{k,ef} \le 1.2, 2.5 \le m_k \le 3.0$. all frequencies are discrete, with an accuracy of 0.01 GHz. A_s is set from 0.7 to 1 with a step size of 0.01. Note that for a given type of application in a heterogeneous system, it only needs to run once to fix the energy allocation factor α. Because the parameter range of the experiment is fixed, a large number of experimental results show a relatively stable law, so the result data of the following specific experiments are only randomly selected once to display.

5.2. Experimental results for FFT applications

We use the parameter ρ as the size of the FFT parallel application, and the total number of tasks is $|T| = (2 \times \rho - 1) + \rho \times \log_2\rho$, where $\rho = 2^y$ [28]. Fig. 4 shows an example of an FFT parallel application with $\rho = 4$.

Experiment 1. This experiment uses the FFT application graph, and experiment with the same size FFT graph under different energy constraints, and compare the energy consumption value and schedule length is generated by different scheduling algorithms. The size of the FFT graph is set to $\rho = 64$, that is, the total number of tasks is 511. For comparison, the energy consumption value of the HEFT algorithm scheduling result is taken as the standard here, denoted as $E_{\text{HEFT}}(G)$, and the energy constraint range is set to $E_{\text{HEFT}}(G)\times0.3$ to $E_{\text{HEFT}}(G)\times0.8$, that is, from strict energy constraints to loose energy constraints.

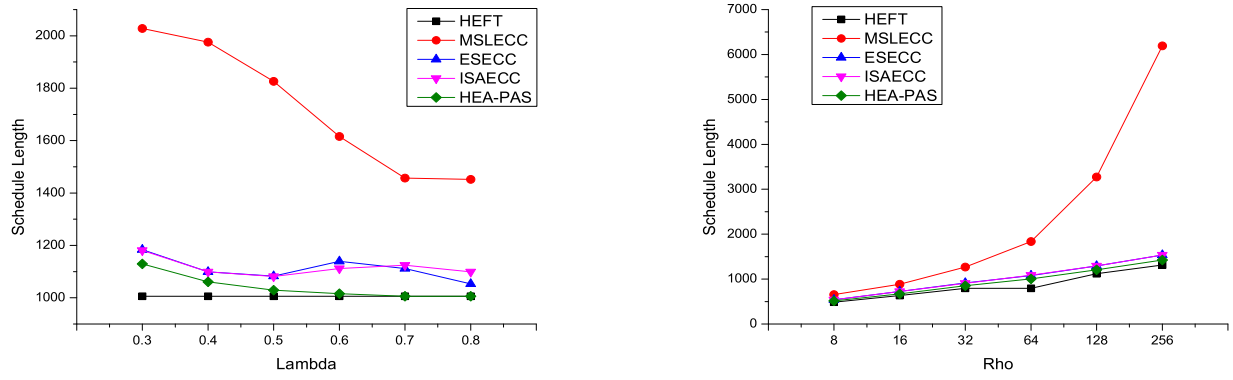
In particular, when the constraints are more stringent, that is, less energy is given, the improvement of HEA-PAS is greater than that of MSLECC, and the performance of ESECC and ISAECC is also improved to a certain extent. For example, as shown in Table 6, when the energy

Table 6
Final energy consumption (Unit: kWs) and actual schedule length (Unit: s) of FFT applications by fix $\rho = 64$ for varying energy constraints.

λ	$\|T\|$	$E_{cons}(G)$	HEFT [28]		MSLECC [16]		ESECC [17]		ISAECC [18]		HEA-PAS	
			$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
0.3	511	3569.98	11899.92	1006.00	3569.98	2027.92	3569.86	1183.46	3569.86	1180.63	3569.88	**1129.24**
0.4	511	4759.97	11899.92	1006.00	4759.97	1976.05	4759.85	1099.29	4759.96	1098.71	4759.88	**1061.14**
0.5	511	5949.96	11899.92	1006.00	5949.96	1826.20	5949.91	1082.59	5948.30	1081.25	5949.79	**1029.36**
0.6	511	7139.95	11899.92	1006.00	7139.95	1615.83	7132.86	1139.48	7130.77	1112.00	7138.976	**1015.80**
0.7	511	8329.95	11899.92	1006.00	8329.94	1457.01	8320.70	1112.00	8311.83	1124.78	8315.85	**1006.00**
0.8	511	9519.94	11899.92	1006.00	9519.94	1452.00	9477.15	1052.88	9508.99	1098.93	9515.31	**1006.00**

Table 7
Final energy consumption (Unit:kWs) and actual schedule length (Unit:s) of FFT application by fix $E_{cons}(G) = E_{HEFT}(G) \times 0.5$ for varying different ρ.

ρ	$\|T\|$	$E_{cons}(G)$	HEFT [28]		MSLECC [16]		ESECC [17]		ISAECC [18]		HEA-PAS	
			$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
8	39	489.07	978.14	484.80	489.07	653.88	480.84	541.16	480.44	540.42	485.24	**513.45**
16	95	1214.21	2428.43	634.20	1214.21	885.75	1206.32	724.65	1204.51	724.11	1213.35	**669.26**
32	223	2580.16	5160.31	796.40	2580.16	1266.88	2578.51	913.50	2578.55	907.24	2579.64	**852.46**
64	511	5691.31	11382.61	939.20	5691.31	1836.68	5690.28	1079.93	5690.35	1085.15	5691.01	**1003.92**
128	1151	11994.07	23988.14	1121.00	11994.07	3294.27	11993.98	1292.00	11993.91	1291.98	11993.95	**1207.79**
256	2559	23722.07	47444.14	1313.50	23722.07	6191.44	23721.98	1539.25	23721.92	1537.49	23721.99	**1424.50**



(a) Different energy constraints (λ). (b) Different application scales (ρ).

Fig. 5. The results of FFT parallel applications under different energy consumption constraints and application scales.

constraint is set to $\lambda = 0.3$, the schedule length generated by HEA-PAS is 1129.24, while the schedule length of MSLECC is 2027.92, and the schedule length of the state-of-the-art algorithm ISAECC is 1180.63. Compared with MSLECC, the schedule length of HEA-PAS is reduced by 898.68. Compared with ISAECC, it is also reduced by 51.39. Fig. 5(a) plots the changing trend of the final schedule length. The reason for this reduction is that when the energy is relatively small, using the energy allocation method in the MSLECC algorithm will only allocate high-priority tasks, while low-priority tasks will not have enough energy. The ISAECC algorithm uses a weighting strategy for energy consumption levels, but does not consider the difference in energy consumption requirements between high priority and low priority.

Experiment 2. In this experiment, we fix $\lambda = 0.5$ (i.e., $E_{cons}(G) = E_{HEFT}(G) \times 0.5$), and then we change the FFT application from a small scale ($\rho = 8, |T| = 39$) to a large scale ($\rho = 256, |T| = 2559$). The scheduling results are shown in Table 7. Similar to Experiment 1, although HEFT can obtain the minimum schedule length, HEFT does not consider energy consumption and consumes more energy in each case. Moreover, MSLECC, ESECC, ISAECC, and HEA-PAS can always meet energy consumption limits. Furthermore, Fig. 5(b) plots the changing trend of the final schedule length. As can be seen from the figure, for the MSLECC algorithm, with the scale grows, the final schedule length will increase sharply, while the final schedule length obtained by our algorithm HEA-PAS will only increase slightly, which fundamentally explains our algorithm has better scalability. We also show advantages compared with ESECC and ISAECC algorithms.

5.3. Experimental results for GE applications

In this section, another important real parallel application, GE application, is used as an experimental object to verify the performance of the proposed HEA-PAS algorithm. While, the number of tasks can be calculated by $|T| = \frac{\rho^2+\rho-2}{2}$. Fig. 6 shows an example of a GE parallel application with $\rho = 5$.

Experiment 3. This experiment uses GE applications, and conducts different energy constraint experiments on GE graphs of the same scale. Then, we compares the energy consumption values and schedule lengths generated by different scheduling algorithms. Here, the size of GE application is set to $\rho = 31$, that is, the total number of tasks is 495. This is to get closer to the number of tasks in the FFT mapping in Experiment 1, so that the level comparison can be approximated. Parallel effect of two application graphs. Similarly, the energy consumption value of the HEFT algorithm scheduling result here is still the standard, and the energy constraint range $E_{cons}(G)$ is set to $E_{HEFT}(G) \times \lambda$. We change λ from 0.3 to 0.8. Fig. 7(a) plots the final schedule length in all cases, and the detailed scheduling results are shown in Table 8. Both indicate that HEA-PAS has less schedule length than MSLECC and ISAECC.

Although the schedule length of the Gaussian elimination application of the same scale is higher than that of the FFT application, their overall trends are similar. First, as the budget increases, the completion time of applications generated using HEA-PAS gradually decreases. The results show that the proposed HEA-PAS algorithm is effective in different types of parallel applications, and the effect is stable.

Table 8
Final energy consumption (Unit: kWs) and actual schedule length (Unit: s) of GE applications by fix $\rho = 31$ for varying energy constraints.

λ	$\|T\|$	$E_{cons}(G)$	HEFT [28]		MSLECC [16]		ESECC [17]		ISAECC [18]		HEA-PAS	
			$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
0.3	495	3026.88	10089.60	3188.00	3026.88	5794.8	3026.81	4116.25	3026.82	4117.23	3026.80	**3996.12**
0.4	495	4035.84	10089.60	3188.00	4035.84	5517.97	4035.70	3896.38	4035.72	3899.87	4035.79	**3808.25**
0.5	495	5044.8	10089.60	3188.00	5044.80	5159.45	5044.62	3783.16	5044.69	3778.35	5044.80	**3726.82**
0.6	495	6053.76	10089.60	3188.00	6053.76	5083.72	6051.63	3684.74	6052.30	3683.53	6053.67	**3600.70**
0.7	495	7062.72	10089.60	3188.00	7062.72	4338.14	7058.58	3652.19	7059.27	3649.09	7061.96	**3520.45**
0.8	495	8071.68	10089.60	3188.00	8071.68	4121.34	8059.30	3426.84	8066.36	3397.39	8066.29	**3319.90**

Table 9
Final energy consumption (Unit: kWs) and actual schedule length (Unit: s) of GE application by fix $E_{cons}(G) = E_{HEFT}(G) \times 0.5$ for varying different ρ.

ρ	$\|T\|$	$E_{cons}(G)$	HEFT [28]		MSLECC [16]		ESECC [17]		ISAECC [18]		HEA-PAS	
			$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
13	90	1089.43	2178.86	1317.00	1089.43	2036.43	1083.51	1697.66	1083.62	1703.6	1089.27	**1635.05**
21	230	2265.30	4530.60	2172.00	2265.30	3265.31	2265.10	2619.71	2265.16	2617.55	2265.16	**2511.19**
31	495	5280.95	10561.89	3490.00	5280.94	5334.57	5280.77	4142.10	5280.73	4141.99	5280.78	**3966.62**
47	1127	12290.43	24580.85	5236.00	12290.42	8402.05	12287.70	6098.55	12288.12	6116.46	12288.71	**6035.58**
71	2555	30858.79	61717.57	7763.00	30858.78	13258.22	30853.52	9043.57	30853.37	9064.61	30856.62	**8901.57**

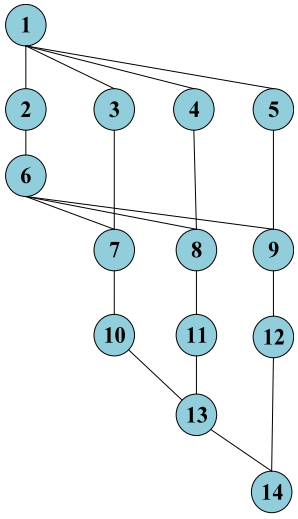


Fig. 6. Example of the GE parallel application with $\rho = 5$.

Experiment 4. In this experiment, we fix $E_{cons}(G)$ as $E_{HEFT}(G) \times 0.5$, and change ρ from 13 ($|N| = 90$, small scale) to 71 ($|T| = 2555$, large scale). Corresponding to parallel FFT applications, these ratios are roughly equal to the number of tasks in Experiment 2. Similarly, the actual energy consumption of MSLECC, ESECC, ISAECC and HEA-PAS is still within the given constraints. Compared with the MSLECC algorithm, our algorithm can produce a shorter schedule length. In addition, through experimental comparison, when ρ increases, the schedule length obtained by our method and HEFT only slightly increases in the FFT application experiment, but it increases sharply in the GE experiment. At the same time, the schedule length generated by MSLECC has also increased dramatically. This phenomenon is due to the fact that FFT parallel applications have better parallelism than GE parallel applications.

Fig. 7(b) shows the variation curve of the schedule length applied by GE under different energy constraints. Table 9 lists the detailed data values of these 5 algorithms. By comparing with Table 7, it can be found that under the same energy constraint, the schedule length of GE application is longer than that of FFT application, which is about 3 times different. This also verifies that the parallelism of the FFT application is much higher than that of the GE application. At the same time, the comparison between HEA-PAS and MSLECC also shows that HEA-PAS has better performance, and can also produce a smaller schedule length compared with other proposed algorithms.

5.4. Randomly generated parallel applications

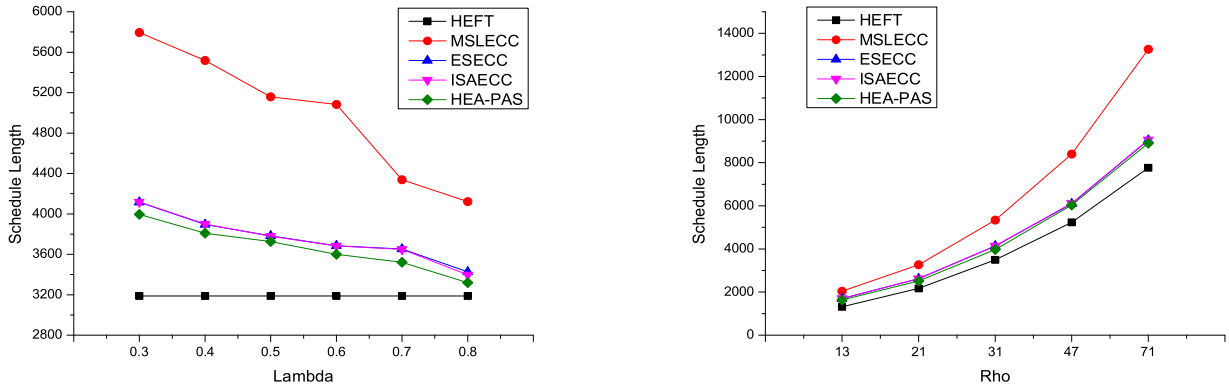
Without loss of generality, we consider parallel applications randomly generated by the task graph generator [37]. It is assumed that the target platform is composed of heterogeneous distributed systems. As long as the value of the heterogeneity factor is adjusted, it is easy to realize the heterogeneity of randomly generated parallel applications. The number of randomly generated tasks is roughly equal to that of FFT and GE parallel applications.

Experiment 5. To prevent other conditions from having too much influence on the HEA-PAS algorithm and the MSLECC algorithm, we set the number of tasks is fixed to $|N| = 511$, heterogeneity factor $\hbar = 0.1$ and the energy constraint range $E_{cons}(G)$ is set to $E_{HEFT}(G) \times \lambda$. We change λ from 0.3 to 0.8. Fig. 8(a) plots the final schedule length information of parallel applications randomly generated with low heterogeneity under different energy consumption constraints by using five different algorithms. It can be seen that the performance of the HEA-PAS algorithm is better than that of MSLECC and other algorithms proposed in previous studies, whether it is a low heterogeneity application or a high heterogeneity application.

Experiment 6. As in Experiment 5, to prevent other conditions from having too much influence on the HEA-PAS algorithm and the MSLECC algorithm, we set the number of tasks is fixed to $|T| = 511$, heterogeneity factor $\hbar = 1.0$ (high heterogeneity) and the energy constraint range $E_{cons}(G)$ is set to $E_{HEFT}(G) \times \lambda$. We change λ from 0.3 to 0.8.

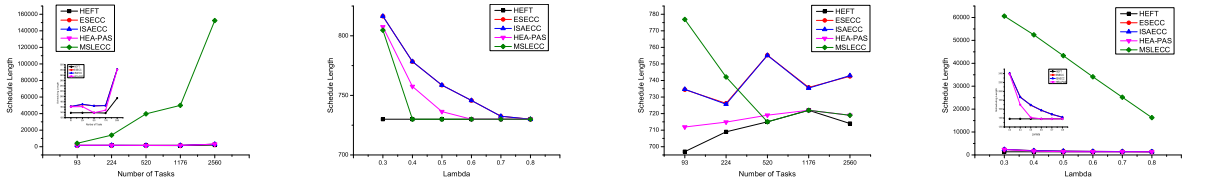
Fig. 8(b) plots the final schedule length information of a parallel application randomly generated with low heterogeneity under different energy consumption constraints by using five different algorithms. Since the target computing platform is composed of heterogeneous processors, the degree of heterogeneity may also affect application performance. Combining Experiment 5 and Experiment 6, it can be seen that the performance of HEA-PAS algorithm is better than that of MSLECC and other algorithms proposed in previous studies, whether it is low heterogeneity applications or high heterogeneity applications.

Experiment 7. In this experiment, we observe the performance of the algorithms in low heterogeneity applications at different scales. We set heterogeneity factor $\hbar = 0.1$ (low heterogeneity), and select the number of tasks from {93, 225, 520, 1175, 2560}. The value of $E_{cons}(G)$ remains unchanged, and set to $E_{HEFT}(G) \times 0.5$. Table 10 shows the results of scheduling different numbers of tasks through parallel applications generated randomly with low heterogeneity when using all



(a) Different energy constraints (λ). (b) Different application scales (ρ).

Fig. 7. Comparison results of GE parallel applications under different energy consumption constraints and application scales.



(a) Different tasks of low-heterogeneity parallel applications. (b) Different λ for low-heterogeneity parallel applications. (c) Different tasks of high-heterogeneity applications. (d) Different λ for high-heterogeneity parallel applications.

Fig. 8. The scheduling results of low and high heterogeneity parallel applications.

Table 10
Final energy consumption (Unit: kWs) and actual schedule length (Unit: s) of low heterogeneity ($\hbar = 0.1$) applications by fix $|T| = 551$ for varying energy constraints.

λ	$\|T\|$	$E_{\text{cons}}(G)$	HEFT [28]		MSLECC [16]		ESECC [17]		ISAECC [18]		HEA-PAS	
			$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
0.3	551	7304.43	24348.11	1382.00	7304.43	60606.71	7304.26	2401.48	7304.06	2405.62	7304.38	**2385.31**
0.4	551	9739.25	24348.11	1382.00	9739.24	52418.92	9738.96	1875.08	9738.88	1870.19	9738.94	**1698.88**
0.5	551	12174.06	24348.11	1382.00	12174.05	43324.98	12173.75	1688.70	12173.64	1692.30	12174.04	**1412.33**
0.6	551	14608.87	24348.11	1382.00	14608.87	34108.90	14608.60	1574.54	14608.73	1574.94	14608.44	**1382.00**
0.7	511	17043.68	24348.11	1382.00	17043.68	25184.08	17043.67	1484.62	17043.41	1484.09	17043.49	**1382.00**
0.8	511	19478.49	24348.11	1382.00	19478.49	16288.37	19478.31	1415.83	19478.28	1419.29	19478.06	**1382.00**

Table 11
Final energy consumption (Unit: kWs) and actual schedule length (Unit: s) of low heterogeneity ($\hbar = 1.0$) applications by fix $|T| = 551$ for varying energy constraints.

λ	$\|T\|$	$E_{\text{cons}}(G)$	HEFT [28]		MSLECC [16]		ESECC [17]		ISAECC [18]		HEA-PAS	
			$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
0.3	551	781.28	2604.25	730.00	781.27	**804.72**	781.27	816.50	781.27	816.90	781.27	807.53
0.4	551	1041.70	2604.25	730.00	1041.70	**730.00**	1041.63	778.41	1041.69	778.26	1041.70	757.47
0.5	551	1302.13	2604.25	730.00	1302.12	730.00	1301.41	758.58	1301.49	758.34	1302.07	**736.43**
0.6	551	1562.55	2604.25	730.00	1562.55	730.00	1555.87	745.75	1560.98	745.21	1562.54	**730.00**
0.7	511	1822.98	2604.25	730.00	1822.97	730.00	1820.02	732.51	1811.67	732.39	1822.95	**730.00**
0.8	511	2083.40	2604.25	730.00	2083.40	730.00	2064.25	730.00	2047.80	730.00	2081.85	**730.00**

algorithms. The experimental results show that the proposed HEA-PAS algorithm also has good performance in low heterogeneity applications, and demonstrates the high performance and stability of the proposed algorithm.

Experiment 8. In this experiment, to observe the performance of the proposed algorithm under different scales with high heterogeneity applications, we select the number of tasks from {93, 225, 520, 1175, 2560}, and set heterogeneity factor $\hbar = 1.0$ (high heterogeneity). The value of $E_{\text{cons}}(G)$ remains unchanged, and set to half of $E_{\text{HEFT}}(G)$. Table 11 shows the results of parallel applications that are randomly generated with high heterogeneity for different numbers of tasks with different algorithms. It can be concluded that applications with high heterogeneity may have higher energy savings and shorter schedule length. The experimental results show that the proposed HEA-PAS algorithm shows effectiveness in both low heterogeneity and high heterogeneity, which fully proves the performance advantages and stability of the algorithm.

6. Conclusion

In this study, we solved the problem of minimize the schedule length of energy-constrained parallel applications on HCS. We defined the concepts of task energy demand rate and energy distribution factor to allocate energy of tasks reasonably. We proposed a novel two-stage hybrid energy allocation (HEA) strategy, which divide the allocatable energy into two parts according to the energy allocation factor, namely static pre-allocate energy (SAE) (to meeting low-priority energy requirements) and dynamic pre-allocate energy (DAE) (to meeting high-priority tasks energy requirements), and provided a rigorous

mathematical proof to verify its feasibility. Moreover, we designed a novel scheduling algorithm (HEA-PAS) based on the above two-stage hybrid energy allocation strategy. Extensive experiments conducted in real-world applications and randomly generated applications proved that our algorithm can obtain better schedule length while meeting energy consumption constraints compare with the state-of-the-art algorithms, which is effective and competitive.

In the future, we plan to use the idea of HEA-PAS to find a reasonable energy allocation solution, considering the problem of reliability maximization, and the trade-off between schedule length and reliability under energy constraints.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors wish to express their sincere appreciation to all the anonymous reviewers and the editor for their worthwhile and constructive comments. The research was partially funded by the National Key R&D Program of China (2020YFB2104000), the National Outstanding Youth Science Program of National Natural Science Foundation of China (61625202), the National Natural Science Foundation of China (Grant Nos. 61860206011, 61876061, 62172151).

References

[1] H. Djigal, J. Feng, J. Lu, J. Ge, IPPTS: an efficient algorithm for scientific workflow scheduling in heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 32 (5) (2021) 1057–1071.

[2] Y.C. Lee, A.Y. Zomaya, Energy conscious scheduling for distributed computing systems under different operating conditions, IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 1374–1381.

[3] K. Li, X. Tang, K. Li, Energy-efficient stochastic task scheduling on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 2867–2876.

[4] L. Zhang, K. Li, C. Li, K. Li, Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems, Inform. Sci. 379 (2017) 241–256.

[5] L. Zhang, L. Zhou, A. Salah, Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments, Inform. Sci. 531 (2020) 31–46.

[6] M.D. Weiser, B.B. Welch, A.J. Demers, S. Shenker, Scheduling for reduced CPU energy, in: Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation, OSDI, Monterey, California, USA, November 14–17, 1994, USENIX Association, 1994, pp. 13–23.

[7] B. Salami, H. Noori, M. Naghibzadeh, Fairness-aware energy efficient scheduling on heterogeneous multi-core processors, IEEE Trans. Comput. 70 (1) (2021) 72–82.

[8] M. Jarus, S. Varrette, A. Oleksiak, P. Bouvry, Performance evaluation and energy efficiency of high-density hpc platforms based on intel, AMD and ARM processors, in: European Conference on Energy Efficiency in Large Scale Distributed Systems, Vol. 8046, Springer, 2013, pp. 182–200.

[9] S.K. Mishra, D. Puthal, B. Sahoo, S.K. Jena, M.S. Obaidat, An adaptive task allocation technique for green cloud computing, J. Supercomput. 74 (1) (2018) 370–385.

[10] N. Khattar, J. Sidhu, J. Singh, Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques, J. Supercomput. 75 (8) (2019) 4750–4810.

[11] K. Li, Energy-efficient task scheduling on multiple heterogeneous computers: Algorithms, analysis, and performance evaluation, IEEE Trans. Sustain. Comput. 1 (1) (2017) 7–19.

[12] G. Xie, G. Zeng, L. Liu, R. Li, K. Li, High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems, J. Syst. Archit. (2016) 3–14.

[13] Y. Xu, L. Liu, Z. Ding, DAG-aware joint task scheduling and cache management in spark clusters, in: 2020 IEEE International Parallel and Distributed Processing Symposium, IPDPS, New Orleans, la, USA, May 18-22, 2020, IEEE, 2020, pp. 378–387.

[14] S. Shi, Q. Wang, X. Chu, B. Li, A DAG model of synchronous stochastic gradient descent in distributed deep learning, in: 24th IEEE ICPADS, Singapore, December 11-13, IEEE, 2018, pp. 425–432.

[15] J.D. Ullman, NP-complete scheduling problems, J. Comput. System Sci. 10 (3) (1975) 384–393.

[16] X. Xiao, G. Xie, R. Li, K. Li, Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems, in: Trustcom/BigDataSE/ISPA, 2016, pp. 1471–1476.

[17] J. Song, G. Xie, R. Li, X. Chen, An efficient scheduling algorithm for energy consumption constrained parallel applications on heterogeneous distributed systems, in: ISPA/IUCC, 2017, pp. 32–39.

[18] Z. Quan, Z.J. Wang, T. Ye, S. Guo, Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 31 (5) (2020) 1165–1182.

[19] J. Chen, C. Du, P. Han, X. Du, Work-in-progress: Non-preemptive scheduling of periodic tasks with data dependency upon heterogeneous multiprocessor platforms, in: IEEE Real-Time Systems Symposium, RTSS 2019, Hong Kong, SAR, China, December 3-6, 2019, IEEE, 2019, pp. 540–543.

[20] J. Chen, C. Du, F. Xie, B. Lin, Scheduling non-preemptive tasks with strict periods in multi-core real-time systems, J. Syst. Archit. 90 (2018) 72–84.

[21] J. Chen, C. Du, F. Xie, Z. Yang, Schedulability analysis of non-preemptive strictly periodic tasks in multi-core real-time systems, Real Time Syst. 52 (3) (2016) 239–271.

[22] G. Liu, Y. Zeng, D. Li, Y. Chen, Schedule length and reliability-oriented multi-objective scheduling for distributed computing, Soft Comput. 19 (6) (2015) 1727–1737.

[23] P.R. Kumar, K. Santhakumar, S. Palani, An intelligent approach for optimizing energy consumption and schedule length of embedded multiprocessors, J. Intell. Fuzzy Systems 31 (1) (2016) 579–587.

[24] S. Yu, K. Li, Y. Xu, A DAG task scheduling scheme on heterogeneous cluster systems using discrete IWO algorithm, J. Comput. Sci. 26 (2018) 307–317.

[25] Z. Zong, A. Manzanares, X. Ruan, X. Qin, EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters, IEEE Trans. Comput. 60 (3) (2011) 360–374.

[26] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, X. Huang, Enhanced energy-efficient scheduling for parallel applications in cloud, in: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16, 2012, IEEE Computer Society, 2012, pp. 781–786.

[27] G. Xie, G. Zeng, X. Xiao, R. Li, K. Li, Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems, IEEE Trans. Parallel Distrib. Syst. 28 (12) (2017) 3426–3442.

[28] H. Topcuoglu, S. Hariri, M.Y. Wu, Performance-effective and low-complexity task scheduling forheterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.

[29] K. Cao, J. Zhou, Y. Min, T. Wei, M. Chen, Static thermal-aware task assignment and scheduling for makespan minimization in heterogeneous real-time MPSoCs, in: International Symposium on System and Software Reliability, 2016, pp. 111–118.

[30] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, S. Hu, Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT, Future Gener. Comput. Syst. 93 (2019) 278–289.

[31] M.A. Khan, Scheduling for heterogeneous systems using constrained critical paths, Parallel Comput. 38 (4–5) (2012) 175–193.

[32] G. Xie, Z. Gang, R. Li, K. Li, Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing, IEEE Trans. Sustain. Comput. 2 (2) (2017) 1.

[33] B. Zhao, H. Aydin, D. Zhu, Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints, ACM Trans. Des. Autom. Electron. Syst. 18 (2) (2013) 23:1–23:21.

[34] G. Xie, J. Jiang, Y. Liu, R. Li, K. Li, Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems, IEEE Trans. Ind. Inf. 13 (3) (2017) 1068–1078.

[35] H. Arabnejad, J.G. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 682–694.

[36] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw.-Pract. Exp. 41 (1) (2011) 23–50.

[37] taskgraphgen, [Online] https://sourceforge.net/projects/taskgraphgen/.

Jiwu Peng currently working toward the Ph.D. degree in computer science and technology with the College of Information Science and Engineering, Hunan University, Changsha, China

His research interest includes high-performance computing, heterogeneous distributed computing systems, embedded and real-time systems, cloud computing, software engineering and methodology.

Kenli Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with the University of Illinois at Urbana-Champaign from 2004 to 2005.

He is currently a Cheung Kong professor of computer science and technology with Hunan University, the dean of the College of Information Science and Engineering, Hunan University, and the director with the National Supercomputing Center in Changsha.

His major research interests include high-performance computing, parallel and distributed processing, big data management, and cloud computing. He has published more than 260 research papers in international conferences and journals such as the IEEE Transactions on Computers, the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Industrial Informatics, the IEEE Transactions on Cloud Computing, ICPP, ICDCS, etc.

Prof. Li has served on the editorial board of the IEEE Transactions on Computers, IEEE Transactions on Sustainable Computing, and IEEE Transactions on Industrial Informatics. He is an outstanding member of the CCF and a senior member of the IEEE.

Jianguo Chen received the Ph.D. degree in Computer Science and Technology from Hunan University, China, in 2018. He was a visiting Ph.D. student at the University of Illinois at Chicago from 2017 to 2018. He is currently a Postdoctoral Fellow in the University of Toronto, Canada, and Hunan University, China. His major research areas include distributed computing, machine learning, deep learning, and intelligence transportation systems.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor with Hunan University, China.

His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, computer networking, high-performance computing, big data computing, machine learning, heterogeneous computing systems, embedded systems and cyber-physical systems, CPU-GPU hybrid and cooperative computing, computer architectures and systems, intelligent and soft computing. He has authored or coauthored more than 800 journal articles, book chapters, and refereed conference papers, and has received several best paper awards

He has chaired many international conferences. He is currently an associate editor of the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.