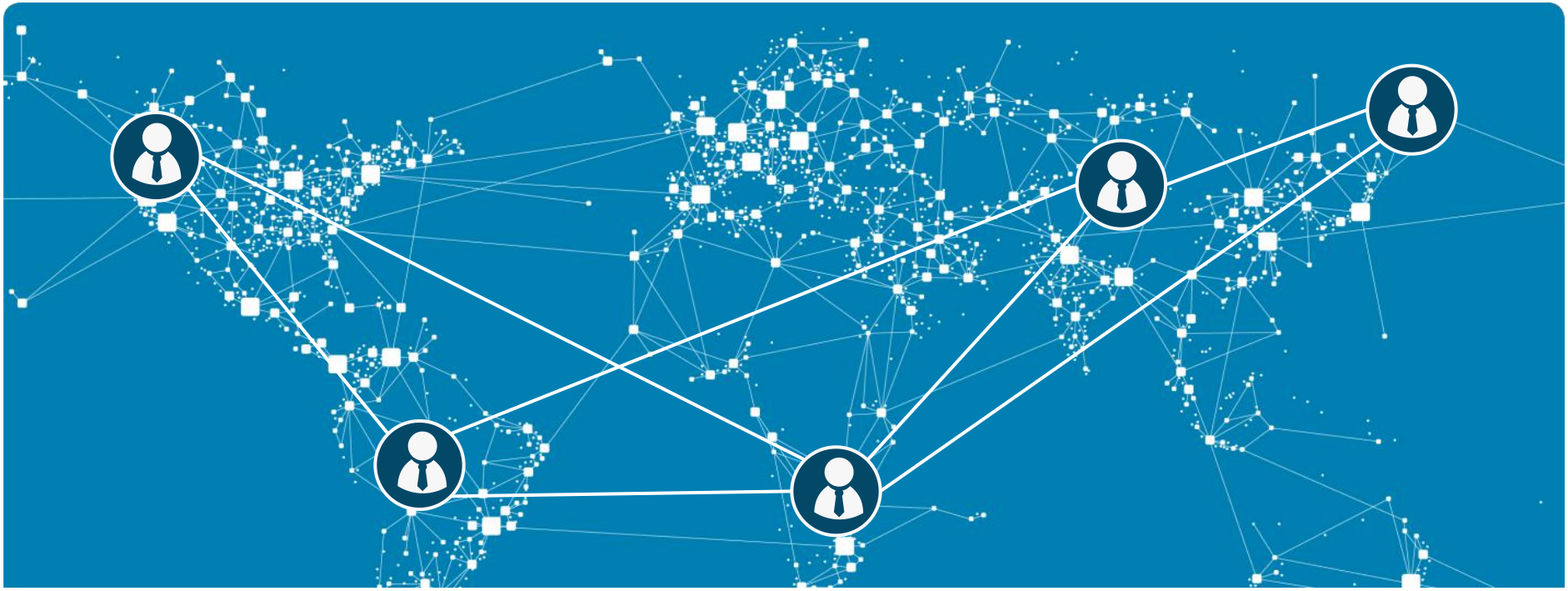


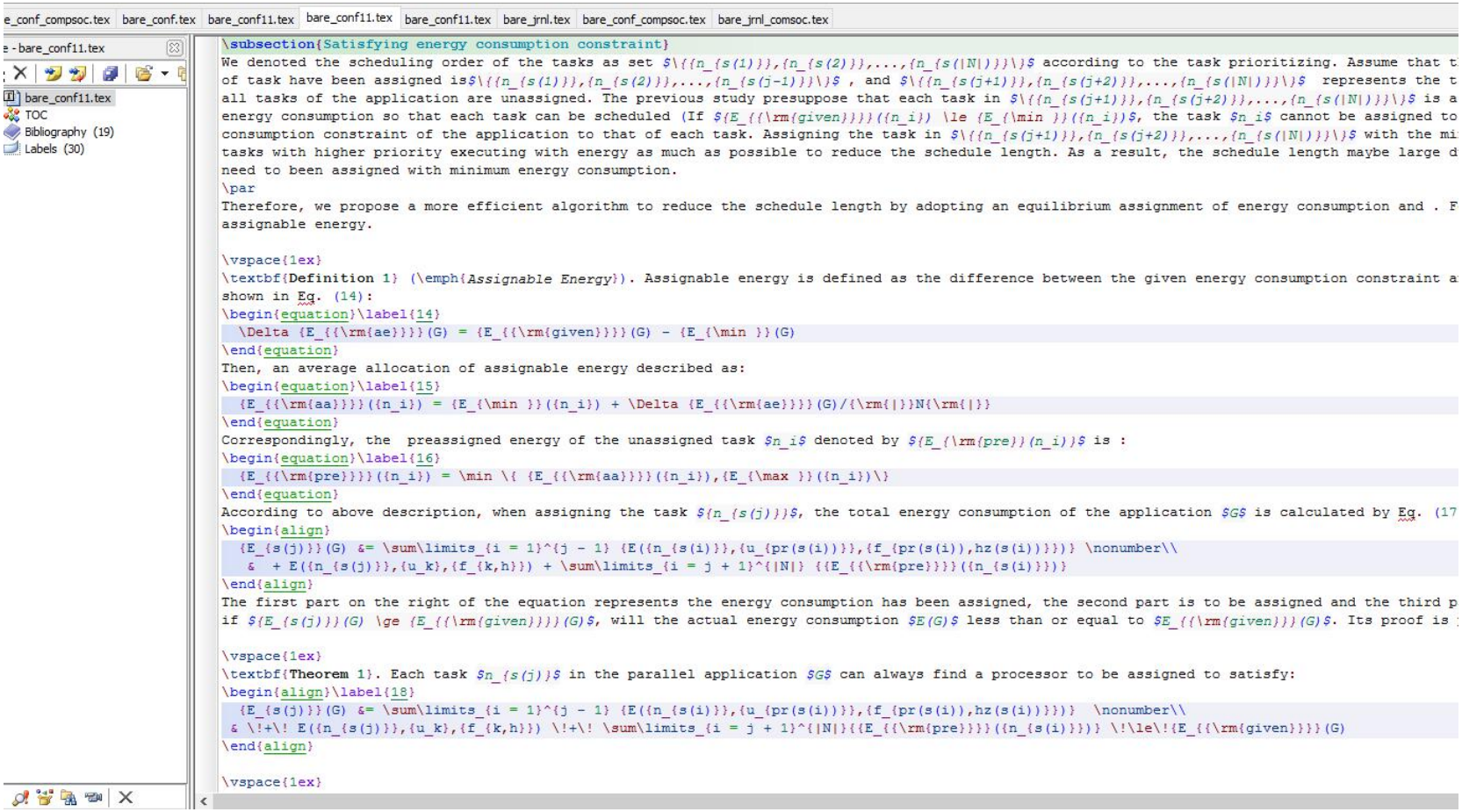
周报告

宋金林　导师：李仁发教授

当前进度

- 基本掌握了Latex排版。
- 论文修改了一遍，准备按照会议的要求作最后的修改。
- 论文创新点较小，还是决定投ISPA，时间上也比较合适（7月15截稿）

Latex的学习


e_conf_compsoc.tex
bare_conf.tex
bare_conf11.tex
bare_conf11.tex
bare_conf11.tex
bare_jrnl.tex
bare_conf_compsoc.tex
bare_jrnl_comsoc.tex
e - bare_conf11.tex
bare_conf11.tex
TOC
Bibliography (19)
Labels (30)
\subsection{Satisfying energy consumption constraint}
We denoted the scheduling order of the tasks as set $\{{n_{s(1)}},{n_{s(2)}},...,{n_{s(|N|)}}\}$ according to the task prioritizing. Assume that t
of task have been assigned is$\{{n_{s(1)}},{n_{s(2)}},...,{n_{s(j-1)}}\}$, and $\{{n_{s(j+1)}},{n_{s(j+2)}},...,{n_{s(|N|)}}\}$ represents the t
all tasks of the application are unassigned. The previous study presuppose that each task in $\{{n_{s(j+1)}},{n_{s(j+2)}},...,{n_{s(|N|)}}\}$ is a
energy consumption so that each task can be scheduled (If ${E_{{\rm given}}}({n_i}) \le {E_{\min }}({n_i})$, the task n_i cannot be assigned to
consumption constraint of the application to that of each task. Assigning the task in $\{{n_{s(j+1)}},{n_{s(j+2)}},...,{n_{s(|N|)}}\}$ with the mi
tasks with higher priority executing with energy as much as possible to reduce the schedule length. As a result, the schedule length maybe large d
need to been assigned with minimum energy consumption.
\par
Therefore, we propose a more efficient algorithm to reduce the schedule length by adopting an equilibrium assignment of energy consumption and . F
assignable energy.
\vspace{1ex}
\textbf{Definition 1} (\emph{Assignable Energy}). Assignable energy is defined as the difference between the given energy consumption constraint a
shown in Eq. (14):
\begin{equation}\label{14}
\Delta {E_{{\rm ae}}}(G) = {E_{{\rm given}}}(G) - {E_{\min }}(G)
\end{equation}
Then, an average allocation of assignable energy described as:
\begin{equation}\label{15}
{E_{{\rm aa}}}({n_i}) = {E_{\min }}({n_i}) + \Delta {E_{{\rm ae}}}(G)/{\rm |}N{\rm |}
\end{equation}
Correspondingly, the preassigned energy of the unassigned task n_i denoted by ${E_{\rm pre}}(n_i)$ is :
\begin{equation}\label{16}
{E_{{\rm pre}}}({n_i}) = \min \{ {E_{{\rm aa}}}({n_i}),{E_{\max }}({n_i})\}
\end{equation}
According to above description, when assigning the task ${n_{s(j)}}$, the total energy consumption of the application G is calculated by Eq. (17
\begin{align}
{E_{s(j)}}(G) &= \sum\limits_{i = 1}^{j - 1} {E({n_{s(i)}},{u_{pr(s(i))}},{f_{pr(s(i)),hz(s(i))}})} \nonumber\\
& + E({n_{s(j)}},{u_k},{f_{k,h}}) + \sum\limits_{i = j + 1}^{|N|} {{E_{{\rm pre}}}({n_{s(i)}})}
\end{align}
The first part on the right of the equation represents the energy consumption has been assigned, the second part is to be assigned and the third p
if ${E_{s(j)}}(G) \ge {E_{{\rm given}}}(G)$, will the actual energy consumption $E(G)$ less than or equal to $E_{{\rm given}}(G)$. Its proof is
\vspace{1ex}
\textbf{Theorem 1}. Each task $n_{s(j)}$ in the parallel application G can always find a processor to be assigned to satisfy:
\begin{align}\label{18}
{E_{s(j)}}(G) &= \sum\limits_{i = 1}^{j - 1} {E({n_{s(i)}},{u_{pr(s(i))}},{f_{pr(s(i)),hz(s(i))}})} \nonumber\\
& \!+\! E({n_{s(j)}},{u_k},{f_{k,h}}) \!+\! \sum\limits_{i = j + 1}^{|N|}{{E_{{\rm pre}}}({n_{s(i)}})} \!\le\!{E_{{\rm given}}}(G)
\end{align}
\vspace{1ex}

An Efficient Scheduling Method for Energy Consumption Constrained Parallel Applications on Heterogeneous Systems

Jinlin Song, Guoqi Xie, Renfa Li
College of Computer Science and Electronic Engineering, Hunan University,
Key Laboratory for Embedded and Network Computing of Hunan Province, China
{songjinlin@hnu.edu.cn, xgqman@hnu.edu.cn, lirenfa@hnu.edu.cn }

***Abstract*—As the explosive growth of energy consumption in current heterogeneous parallel and distributed systems, energy consumption constraint have been one of the primary design issues. Minimizing the schedule length while satisfying the energy consumption constraint of an application is one of the most important problem which have been studied recently. Previous methods tried to presuppose the minimum energy consumption assignment for each task to minimize the schedule length of energy consumption constrained applications on heterogeneous parallel and distributed systems based on the dynamic voltage and frequency scaling (DVFS) technique. However, the preassignment of tasks with the minimum energy consumption does not necessarily lead to the minimization of the schedule length. In this study, we propose an efficient scheduling algorithm using an equilibrium assignment approach to preassign the given energy and select processor for each task. The results of experiments on several real parallel applications validate that the proposed algorithm can obtain shorter schedule lengths while satisfying the energy consumption constraint compared with the state-of-the-art methods in various situations.**

***Index Terms*—dynamic voltage and frequency scaling (DVFS), energy consumption constraint, heterogeneous systems, parallel application, schedule length**

I. Introduction

A. Background

The emergence and development of heterogeneous distributed system is to meet the needs of intensive large-scale scientific computing and commercial computing. Energy consumption management is crucial in the design of modern parallel and distributed system. Vast use of energy will lead to thermal issue which affects the performance of the system itself in reverse [1] and the living environment of people. Therefore, limiting the energy consumption to a certain range while improving the performance of the system is sometimes should to be considered. Many adaptive management techniques have been established to maximize energy efficiency, such as dynamic power management (DPM) and Dynamic voltage and frequency scaling (DVFS). The DPM technology reduce the energy consumption by turning off the idle components and change to the hibernate mode, so it only works when the idle time is long enough. In this study, DVFS technology is adopted, which achieves energy-efficient optimization by simultaneously scaling down the supply voltage and frequency of a processor.

B. Motivation

The problem of minimizing the schedule length of an energy consumption application with precedence-constrained tasks has been solved in a number of studies [2], [8], [12]. However, the first two studies [2], [8] only focus on homogeneous systems. Xiao et al. [12] solved the same problem for heterogeneous systems by presupposing tasks with the minimum energy consumption to satisfy the energy consumption constraint of the application. However, preassigning the minimum energy consumption to each task is not always effective while the given energy consumption is little. The tasks with higher priority having more chance to use the energy of the application, meanwhile, the tasks with low priority have to select the processor with the minimum energy consumption, which will lead to longer schedule length. Therefore, a more efficient scheduling method is needed.

The rest of this paper is organized as follow. Section II reviews related studies. Section III presents related models and preliminaries. Sections IV describe the problem and propose the ESECC algorithms. Section V verifies the ESECC algorithms and Section VI concludes this study.

II. Related works

Energy consumption optimization using DVFS technology has attracted much attention since 1990 [3] when the energy consumption optimization for processors by DVFS was first proposed. In [4], the author studied the problems of minimizing the energy consumption for a single task. In [5], [6], the energy-aware scheduling of independent sequential tasks on a single processor was presented toward energy saving performance while ensuring the QoS guarantee. However, those studies are merely interested in independent task. While the parallel applications with precedence constrained tasks are widely used in high-performance homogeneous/heterogeneous distributed computing systems, some scheduling strategies for those systems have been proposed. The problems of minimizing the expected energy consumption on homogeneous multiprocessor with precedence constrained tasks have been presented in [13], [14], for heterogeneous distributed systems has been studied in [9], [10], [15].

In this paper, we consider a parallel application in heterogeneous distributed computing system. Huang et al. [9] proposed enhanced energy-efficient scheduling (EES) algorithm to reclaim the slack time for each task on the same processor through an upward approach. Moreover, Tang et al. [10] studied the same problem through merging the relatively inefficient processors by reclaiming the slack time to reduce energy dissipation, but it will increases the time complexity. A new method combining downward and upward approaches to implement low time-complexity energy consumption minimization was presented in [15]. However, the aforementioned studies mainly managing the slack time to reduce energy consumption. Those studies mainly consider minimizing the energy consumption while meeting the deadline of the parallel applications. Lee et al. [7] presented two energy-conscious scheduling (ECS and ECS + idle) heuristics to explore the trade off of the schedule length and energy consumption for parallel tasks on heterogeneous distributed systems. Xiao et al. [12] aiming at minimizing the schedule length of a parallel application with energy consumption constrained in heterogeneous distributed systems which is decomposed into two sub-problems, satisfying the energy consumption constraint and minimizing the schedule length. The proposed algorithm (MSLECC) achieves a minimum schedule length while satisfied a given system energy consumption, but the results is too pessimism. In this paper, we study the same problem and propose a fair scheduling algorithm to implement a tight schedule length of energy consumption constrained parallel applications on heterogeneous systems.

III. Models and preliminaries

A. Application model

In this work, we assume that the system consists of a set heterogeneous processors: $U = \{u_1, u_2, ..., u_{|U|}\}$, where $|U|$ denotes the size of set U. A parallel application can be represented by a DAG as shown in Fig. 1. Let $G = \{N, W, M, C\}$. N represents a set of nodes in G: $N = \{n_i\}$, each node means a task which has different worst case execution times (WCETs) on different processors. W is an $|N| \times |U|$ matrix where $w_{i,k}$ represents the WCET of n_i running on the processor u_k with maximum frequency. M is a set of communication edges and each edge $m_{i,j} \in M$ represents the communication message from n_i to n_j. Accordingly, $c_{i,j}$ represents the communication time of $m_{i,j}$ while n_i and n_j are not assigned to the same processor. $pred(n_i)$ represents the set of the immediate predecessor tasks of n_i, and $succ(n_i)$ means the set of its immediate successor tasks. A task without any predecessor is called an entry task n_{entry} (such as n_1 in Fig. 1), and the task that has no successor task is called an exit task n_{exit} (such as n_{10} in Fig. 1).

Considering the example of a DAG shows in Fig. 1. An application consists of ten tasks and executed on three processors. The WCET for each task on different processors with the maximum frequency is shown in Table I. As shown in Fig. 1, the weight 18 on the edge between n_1 and n_2 represents the communication time denoted as $c_{1,2}$. As indicated in Table I, the execution time of n_1 is 14 n_1 executing on u_1 with the maximum frequency. The same task has different execution times on different processors due to the heterogeneity of processors.

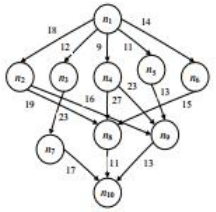


Fig. 1: A simple example of DAG-based parallel application

TABLE I: Execution time matrix 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

B. Energy model

The power consumption of CMOS circuits are consists of static and dynamic energy consumption denoted P_{static} and P_{dynamic} respectively, and the supply voltage and operating frequency for CMOS circuits is nearly linear relationship so that we can change the operating frequency to optimize energy consumption. In this work, we adopt the system-level power model common used in [17]-[19], where the power consumption for a DVFS-enable system with frequency f is given by

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

P_s is the static power and the dynamic power is composed of two parts: P_{ind} and P_{d}. P_{ind} is the frequency-independent dynamic power which is a constant. P_{d} is the frequency-dependent dynamic power which is varies according to the frequency. h represents the system state where $h = 1$ means the system is active, otherwise the system is turned off and $h = 0$. C_{ef} is effective switching capacitance and m is the dynamic power exponent. The static power is unmanageable and is not the main factor, we ignore it in this paper and mainly considering the dynamic part. While we slow down the processing frequencies, the execute time of tasks would increase simultaneously so that lower frequencies may not result in lower energy consumption. Thus, a minimum energy

efficient frequency f_{ee} exists [17]-[19] and it is denoted by

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

Given a processor operating frequency varies from minimum f_{min} to the maximum f_{max}, we can get an interval $[f_{low}, f_{max}]$ where $f_{low} = \max(f_{min}, f_{ee})$. For energy efficiency, the actual frequency f should limit in this interval. The interval is various from the processors since these processors are heterogeneous, so we denote the actual effective frequency set as

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

where $|U|$ represents the size of processors. Similarly, the frequency-independent dynamic power P_{ind} set is $\{P_{1,ind}, P_{2,ind}, ..., P_{|U|,ind}\}$ and the frequency-dependent dynamic power set is $\{P_{1,d}, P_{2,d}, ..., P_{|U|,d}\}$, the effective switching capacitance set is $\{C_{1,ef}, C_{2,ef}, ..., C_{|U|,ef}\}$, the dynamic power exponent set is $\{m_1, m_2, ..., m_{|U|}\}$. Then the energy consumption of ni on the processor u_k with frequency $f_{k,h}$ can be calculated by

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

C. Preliminaries

1) Upward rank value: We use the heterogeneous earliest finish time (HEFT) algorithm to reducing the schedule length to a minimum. The schedule priority of each task is decided by the upward rank value ($ranku$), which is obtained by

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{c_{i,j} + rank_u(n_j)\}, \quad (4)$$

where $\overline{w_i}$ represents the average execution time of task n_i and calculated by $\overline{w_i} = \left(\sum_{k=1}^{|U|} w_{i,k} \right) / |U|$. Table II shows the upward rank values of tasks in Fig. 1. Then the scheduling order of the tasks is $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$.

TABLE II: Upward rank values for tasks in Fig. 1

Task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$rank_u(n_i)$	108	70	80	80	69	63.3	42.7	35.7	44.3	14.7

2) Earliest finish time: Let $EST(n_i, u_k, f_{k,h})$ represent the earliest start time (EST) of task n_i on the processor u_k with frequency $f_{k,h}$, calculated by

$$EST(n_i, u_k, f_{k,h}) = \max \left\{ avail[k], \max_{n_x \in pred(n_i)} \{AFT(n_x) + c'_{x,i}\} \right\}. \quad (5)$$

where $avail[k]$ is the earliest available time when processor u_k is ready for task execution, $AFT(n_x)$ is the actual finish time (AFT) of task n_x, and $c'_{x,i}$ represents the communication time between n_x and n_i. $c'_{x,i}$=$c_{x,i}$ only when n_x and n_i are assigned to the different processor, otherwise,$c'_{x,i}$=0. Then the earliest finish time (EFT) is calculated by

$$EFT(n_i, u_k, f_{k,h}) = EST(n_i, u_k, f_{k,h}) + w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}} \quad (6)$$

The value of $EFT(n_i, u_k, f_{k,h})$ is the task assignment criterion because it can obtain the current shortest schedule length of the application.

IV. Minimizing Schedule Length with Energy Consumption Constraints

A. Problem definition

Assume there is a parallel application G and a heterogeneous processor set U that support different frequency levels. This study addresses the task scheduling problem of determining the processor and energy assignments for each task, and achieve a minimum schedule length while ensuring that the total energy consumption of the application does not exceed the energy consumption constraint. The given energy consumption constraint of G is denoted as $E_{\text{given}}(G)$. The schedule length of the application G is

$$SL(G) = \min_{u_k \in U} \{ \min_{f_{k,h} \in [f_{k,\text{low}}, f_{k,\max}]} \{EFT(n_{\text{exit}}, u_k, f_{k,h})\}\}. \quad (7)$$

The energy consumption constraint is

$$E(G) = \sum_{i=1}^{|N|} E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leq E_{\text{given}}(G). \quad (8)$$

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

$$f_{pr(i),\text{low}} \leq f_{pr(i),hz(i)} \leq f_{pr(i),\max}$$

for $\forall i : 1 \leq i \leq |N|, u_{pr(i)} \in U$.

Assume that the available frequencies of each processor and the execution time of each task on each processor is known, so the minimum and maximum energy consumption for each task can be obtained, denoted by $E_{\min}(n_i)$ and $E_{\max}(n_i)$, respectively, and calculated by Eqs. (9) and (10).

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

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

Thus, the total energy consumption of the application G can be calculated by Eqs. (11) and (12)

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

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

In this study, we assume that the given energy consumption constraint of G is satisfied

$$E_{\min}(G) \leq E_{\text{given}}(G) \leq E_{\max}(G) \quad (13)$$

B. Satisfying energy consumption constraint

We denoted the scheduling order of the tasks as set $\{n_{s(1)}, n_{s(2)}, ..., n_{s(|N|)}\}$ according to the task prioritizing. Assume that the current task to be assigned is $n_{s(j)}$, so the set of task have been assigned is$\{n_{s(1)}, n_{s(2)}, ..., n_{s(j-1)}\}$, and $\{n_{s(j+1)}, n_{s(j+2)}, ..., n_{s(|N|)}\}$ represents the task set where the tasks have not been assigned. Initially, all tasks of the application are unassigned. The previous study presuppose that each task in $\{n_{s(j+1)}, n_{s(j+2)}, ..., n_{s(|N|)}\}$ is assigned to the processor and frequency with the minimum energy consumption so that each task can be scheduled (If $E_{\text{given}}(n_i) \leq E_{\min}(n_i)$, the task n_i cannot be assigned to any processor). The main idea is to transfer the energy consumption constraint of the application to that of each task. Assigning the task in $\{n_{s(j+1)}, n_{s(j+2)}, ..., n_{s(|N|)}\}$ with the minimum energy consumption is a greedy idea that let the tasks with higher priority executing with energy as much as possible to reduce the schedule length. As a result, the schedule length maybe large due to the long execute time of low-priority tasks which need to been assigned with minimum energy consumption.

Therefore, we propose a more efficient algorithm to reduce the schedule length by adopting an equilibrium assignment of energy consumption and . For convenience of description, we first define the assignable energy.

Definition 1 (*Assignable Energy*). Assignable energy is defined as the difference between the given energy consumption constraint and the minimum energy consumption of application G, as shown in Eq. (14):

$$\Delta E_{\text{ae}}(G) = E_{\text{given}}(G) - E_{\min}(G) \quad (14)$$

Then, an average allocation of assignable energy described as:

$$E_{\text{aa}}(n_i) = E_{\min}(n_i) + \Delta E_{\text{ae}}(G)/|N| \quad (15)$$

Correspondingly, the preassigned energy of the unassigned task n_i denoted by $E_{\text{pre}}(n_i)$ is :

$$E_{\text{pre}}(n_i) = \min\{E_{\text{aa}}(n_i), E_{\max}(n_i)\} \quad (16)$$

According to above description, when assigning the task $n_{s(j)}$, the total energy consumption of the application G is calculated by Eq. (17):

$$E_{s(j)}(G) = \sum_{i=1}^{j-1} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) + E(n_{s(j)}, u_k, f_{k,h}) + \sum_{i=j+1}^{|N|} E_{\text{pre}}(n_{s(i)}) \quad (17)$$

The first part on the right of the equation represents the energy consumption has been assigned, the second part is to be assigned and the third part is not been assigned. For each task $n_s(j)$, only if $E_{s(j)}(G) \geq E_{\text{given}}(G)$, will the actual energy consumption $E(G)$ less than or equal to $E_{\text{given}}(G)$. Its proof is provided in the following.

Theorem 1. Each task $n_{s(j)}$ in the parallel application G can always find a processor to be assigned to satisfy:

$$E_{s(j)}(G) = \sum_{i=1}^{j-1} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) + E(n_{s(j)}, u_k, f_{k,h}) + \sum_{i=j+1}^{|N|} E_{\text{pre}}(n_{s(i)}) \leq E_{\text{given}}(G) \quad (18)$$

Proof. We use the mathematical induction to prove it. First of all, for the entry task $n_{s(1)}$, all tasks are not assigned to processors, the application G should satisfy its energy consumption constraint:

$$E_{s(1)}(G) = E(n_{s(1)}, u_k, f_{k,h}) + \sum_{i=2}^{|N|} E_{\text{pre}}(n_{s(i)}) \leq E_{\text{given}}(G) \quad (19)$$

According to Eqs. (14), (15), (16) and (19) we have

$$E_{s(1)}(G) = E(n_{s(1)}, u_k, f_{k,h}) + \sum_{i=2}^{|N|} E_{\text{pre}}(n_{s(i)}) \leq E(n_{s(1)}, u_k, f_{k,h}) + \sum_{i=2}^{|N|} E_{\text{aa}}(n_{s(i)}) \quad (20)$$

and

$$E(n_{s(1)}, u_k, f_{k,h}) + \sum_{i=2}^{|N|} E_{\text{aa}}(n_{s(i)}) = E(n_{s(1)}, u_k, f_{k,h}) + \sum_{i=1}^{|N|} E_{\text{aa}}(n_{s(i)}) - E_{\text{aa}}(n_{s(1)}) = E(n_{s(1)}, u_k, f_{k,h}) + E_{\text{given}}(G) - E_{\text{aa}}(n_{s(1)})$$

Because $E_{\text{aa}}(n_{s(1)})$ is larger than or equal to $E_{\min}(n_{s(1)})$, that is, the processor with the minimum energy consumption can be assigned to task $n_{s(1)}$ at the least. Hence, $n_{s(1)}$ can find an assigned energy to satisfy:

$$E(n_{s(1)}, u_k, f_{k,h}) + E_{\text{given}}(G) - E_{\text{aa}}(n_{s(1)}) \leq E_{\text{given}}(G) \quad (21)$$

Then according to Eqs. (20) and (21), the Eq. (19) is satisfied.

Second, assume that the jth task $n_{s(j)}$ can find an assigned processor $u_{pr(s(j))}$ and frequency $f_{pr(s(j)),hz(s(j))}$ to satisfy constraint, and we have:

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

such that,

$$E_{s(j)}(G) = \sum_{i=1}^{j} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) + \sum_{i=j+1}^{|N|} E_{\text{pre}}(n_{s(i)}) \le E_{\text{given}}(G)$$

Hence,

$$\sum_{i=1}^{j} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) \le E_{\text{given}}(G) - \sum_{i=j+1}^{|N|} E_{\text{pre}}(n_{s(i)}) \quad (22)$$

Thus, for the $(j+1)$th task, the energy consumption of an application is

$$E_{s(j+1)}(G) = \sum_{i=1}^{j} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) + E(n_{s(j+1)}, u_k, f_{k,h}) + \sum_{i=j+2}^{|N|} E_{\text{pre}}(n_{s(i)})$$

Acroding to Eq. (22), we have

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

that is,

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

The values of $E_{\text{pre}}(n_{s(j+1)})$ is larger than $E_{\min}(n_{s(j+1)})$, so when $n_{s(j+1)}$ is assigned an energy consumption in range of $E_{\min}(n_{s(j+1)})$ to $E_{\text{pre}}(n_{s(j+1)})$, we have:

$$E_{s(j+1)}(G) \le E_{\text{given}}(G)$$

That means $n_{s(j+1)}$ can also find an assigned processor to satisfy the energy consumption constraint. When all tasks can find individual assigned processors to satisfy the energy consumption constraint, the correctness of **Theorem 1** is proofed.

C. Proposed ESECC algorithm

When assigning the task $n_{s(j)}$, according to **Theorem 1**, it should satisfy

$$E(n_{s(j)}, u_k, f_{k,h}) \le E_{\text{given}}(G) - \sum_{i=1}^{j-1} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) - \sum_{i=j+1}^{|N|} E_{\text{pre}}(n_{s(i)})$$

Hence, let the energy consumption constraint of the task be

$$E_{\text{given}}(n_{s(j)}) = E_{\text{given}}(G) - \sum_{i=1}^{j-1} E(n_{s(i)}, u_{pr(s(i))}, f_{pr(s(i)),hz(s(i))}) - \sum_{i=j+1}^{|N|} E_{\text{pre}}(n_{s(i)}) \quad (23)$$

Then we only need to consider the energy consumption constraint of each task rather than the application G. Because the maximum energy consumption constraint of $n_{s(j)}$ is $E_{\max}(n_{s(j)})$, so the task should satisfy the following constraint

$$E(n_{s(j)}, u_k, f_{k,h}) \le \min\{E_{\text{given}}(n_{s(j)}), E_{\max}(n_{s(j)})\}$$

After transferring the energy consumption constraint of applications to that of each task, tasks are assigned to processors with the minimum EFT by using the insertion-based scheduling strategy while satisfying the energy consumption constraint of tasks. Then we propose the whole algorithm and each step is described in the ESECC algorithm.

Algorithm 1 The ESECC Algorithm

Input: $U = \{u_1, u_2, ..., u_{|U|}\}$, G, $E_{\text{given}}(G)$
Output: $SL(G)$, $E(G)$
1: Sort the tasks in a list *dlist* by descending order of $rank_u$ values;
2: **for all** $(n_i \in N)$ **do**
3: Calculate $E_{\min}(n_i)$ and $E_{\max}(n_i)$ using Eqs. (9) and (10),respectively;
4: **end for**
5: Calculate $E_{\min}(G)$ and $E_{\max}(G)$ using Eqs. (11) and (12), respectively;
6: **for all** $(n_i \in N)$ **do**
7: Calculate $E_{\text{pre}}(n_i)$ using Eqs. (14), (15), (16);
8: **end for**
9: **while** (there are tasks in *dlist*) **do**
10: n_i = *dlist*.out();
11: Calculate $E_{\text{given}}(n_i)$ using Eq. (23);
12: **for** $(\forall k, u_k \in U)$ **do**
13: **for** $(\forall h, f_{k,h} \in [f_{k,\text{low}}, f_{k,\max}])$ **do**
14: Calculate $E(n_i, u_k, f_{k,h})$ using Eq. (3);
15: **if** $(E(n_i, u_k, f_{k,h}) > \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\})$ **then**
16: continue;
17: **end if**
18: Calculate $EFT(n_i, u_k, f_{k,h})$ using Eq. (6);
19: **if** $(EFT(n_i, u_k, f_{k,h}) < AFT(n_i))$ **then**
20: $pr(i) \leftarrow k$;
21: $f_{pr(i),hz(i)} \leftarrow f_{k,h}$;
22: $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leftarrow E(n_i, u_k, f_{k,h})$;
23: $AFT(n_i) \leftarrow EFT(n_i, u_k, f_{k,h})$;
24: **end if**
25: **end for**
26: **end for**
27: **end while**
28: Calculate the actual energy consumption $E(G)$ using Eq. (8);
29: Calculate the schedule length $SL(G) = AFT(n_{\text{exit}})$;
30: **return** $E(G)$ and $SL(G)$.

The main idea is choose a suitable preassigned value of energy consumption for each unassigned task and avoid a unfairness result. The details of ESECC are explained as follows:

1) In line 6-8, we calculate the preassigned energy consumption for each task.

2) In line 12-26, all the processors and frequencies are traversed and we select the processor and frequency with the minimum EFT.

3) In line 28-29, we calculate the actual energy consumption $E(G)$ and schedule length $SL(G)$.

The core schedule idea is HEFT which has a low time complexity of $O(|N|^2 \times |U|)$, so the time complexity of ESECC is $O(|N|^2 \times |U| \times |F|)$. $|F|$ represents the maximum number of discrete frequencies from the lowest to the maximum actual effective frequencies.

D. Example of the ESECC algorithm

Considering the parallel application in Fig. 1, the parameter of each processor is shown in Table III. In this example, the maximum frequency $f_{k,\max}$ for each processor is 1.0 and frequency precision is set to 0.01.

TABLE III: Power and frequency parameters of the processors

u_k	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	m_k	$f_{k,\text{low}}$	$f_{k,\max}$
u_1	0.03	0.8	2.9	0.22	1.0
u_2	0.04	0.8	2.5	0.21	1.0
u_3	0.07	1.0	2.5	0.19	1.0

Therefore, we can obtain the minimum and maximum energy consumption of G according to Eqs. (12) and (13) respectively. We set the energy consumption constraint of G is $E_{\text{given}}(G) = 0.5 \times E_{\max}(G)$. Then we can obtain the scheduling result as shown in Table IV by using ESECC.

TABLE IV: ESECC-generated task assignment of the application 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	8.5499	u_3	0.91	0	9.8901	8.5051
n_3	8.0628	u_1	0.93	21.8901	33.7181	8.0214
n_4	8.1888	u_2	1.00	18.8901	26.8901	6.7200
n_2	9.8414	u_3	0.56	9.8901	42.0330	9.7932
n_5	8.2436	u_2	0.81	26.8901	42.9395	8.2236
n_6	8.3925	u_1	0.86	33.7181	48.8344	8.2622
n_9	9.3174	u_2	0.94	58.0330	70.7990	9.2597
n_7	7.3667	u_1	1.00	48.8344	55.8344	5.8100
n_8	8.5112	u_1	1.00	61.0330	66.0330	4.1500
n_{10}	12.2487	u_2	1.00	77.0330	84.0330	5.8800
$E(G)$=74.6252, $SL(G)$=84.0330						

The final energy consumption of the application is 74.6252, which is lower than $E_{\text{given}}(G)$. Meanwhile, the schedule length is 84.033. Fig. 2 shows the scheduling Gantt chart of the parallel application G in Fig. 1 using ESECC algorithm. Table V shows the final energy consumption and schedule length obtain by using ESECC and MSLECC algorithm respectively. Both the MSLECC and ESECC can meet the constraint of energy consumption, but the ESECC has a shorter schedule length than MSLECC.

TABLE V: Scheduling result of the application in Fig. 1 using MSLECC and ESECC

Algorithm	$E(G)$	$SL(G)$
MSLECC	80.9939	129.3660
ESECC	74.6252	84.0330

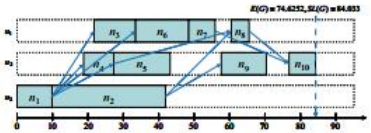


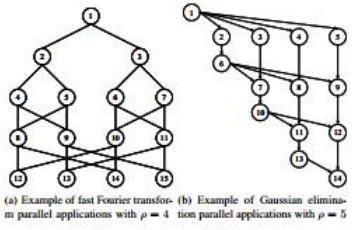
Fig. 2: Scheduling Gantt chart of the parallel application in Fig. 1 using ESECC algorithm.

V. Experiments and Discussion

In this section, we compare the actual energy consumption $E(G)$ and the final schedule length $SL(G)$ with HEFT and MSLECC.

A. Experimental metrics

We resort to simulation method to verify our algorithms. We implement a simulator in Java language. The parameters of the processor and application are similar to MSLECC as follows: $10\text{ms} \le w_{i,k} \le 100\text{ms}$, $10\text{ms} \le c_{i,j} \le 100\text{ms}$, $0.03 \le P_{k,\text{ind}} \le 0.07$, $0.8 \le C_{k,\text{ef}} \le 1.2$, $2.5 \le m_k \le 3.0$, and $f_{k,\max}$=1 GHz. All frequencies are discrete, and the precision is 0.1 GHz. The number of heterogeneous ECU is 64 which values are generated randomly. We use real parallel applications, such as the Gaussian elimination applications and fast Fourier transform applications to verify the effectiveness of the proposed algorithm [16]. Fig. 3(a) shows an example of the fast Fourier transform application with $\rho = 4$ and Fig. 3(b) shows an example of the Gaussian elimination application with $\rho = 5$. ρ is a parameter to define the matrix size of the application, and the task numbers in a FFT graph is $|N| = (2 \times \rho - 1) + \rho \times \log_2 \rho$, where $\rho = 2^y$ for some integer y. The task numbers in a GE graph is $|N| = \frac{(\rho^2+\rho-2)}{2}$.



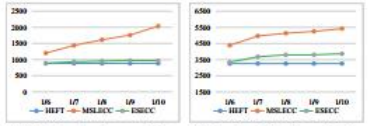
(a) Example of fast Fourier transform parallel applications with $\rho = 4$ (b) Example of Gaussian elimination parallel applications with $\rho = 5$

Fig. 3: Example of real parallel application

B. Varying energy consumption constraint

We first consider the parallel applications with different energy consumption constraints, two experiments are done. The details are described as follow:

Experiment 1. In this experiment, we compare the actual energy consumption and final schedule length of a fast Fourier transform application for varying energy consumption constraints. The matrix size is set to $\rho = 64$ (i.e., $|N|$=511) , and the energy consumption constraint is varying from $(E_{\min}(G) + E_{\max}(G))/6$ to $(E_{\min}(G) + E_{\max}(G))/10$.



(a) fast Fourier transform application (b) Gaussian elimination application

Fig. 4: Final schedule length for varying $E_{\text{given}}(G)$

Fig. 4(a) shows the final schedule length value of fast Fourier transform application on different energy consumption constraints, and the detail values are shown in Table VI. We can see the actual energy consumption using both ESECC and MSLECC algorithm are not exceed the given energy consumption constraint in all case. The proposed ESECC algorithm has a better performance on schedule length than MSLECC, and the values are approximate to HEFT. Specially, ESECC can obviously minimizing schedule length compared with MSLECC while the total given energy consumption is little. As shown in Table VI, When the given energy is limited to $(E_{\min}(G) + E_{\max}(G))/10$, the schedule length using ESECC is 963.37 while it is 2034.57 using MSLECC. ESECC is better than MSLECC by 52.6%. The reason is that the low priority tasks with not enough energy available using MSLECC and need to execute with minimum energy consumption, thus it will lead to longer schedule length. It only minimize the execute time of the tasks near the entry task.

TABLE VI: Scheduling results of fast Fourier transform applications with ρ = 64(i.e., $|N|$ = 511) for varying energy consumption constraints

$E_{\text{given}}(G)$	HEFT		MSLECC		ESECC	
	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
10011.79	11097.05	883	10011.78	1202.48	9970.09	887.61
8581.53	11097.05	883	8581.53	1437.30	8551.20	932.24
7508.84	11097.05	883	7508.84	1613.30	7506.98	952.89
6674.52	11097.05	883	6674.52	1761.24	6670.00	969.48
6007.07	11097.05	883	6007.07	2034.57	6006.95	963.37

Experiment 2. In this experiment, we compare the actual energy consumption and final schedule length of a Gaussian elimination application for varying energy consumption constraints. The matrix size is set to $\rho = 32$ (i.e., $|N|$=527). The total number of task for the Gaussian elimination application is similar to that of the fast Fourier transform application, and the energy consumption constraint is varying from $(E_{\min}(G)+E_{\max}(G))/6$ to $(E_{\min}(G) + E_{\max}(G))/10$.

Fig. 4(b) shows the final schedule length value of Gaussian elimination application on different energy consumption constraints, and the detail values are shown in Table VII. As indicated in Table VII, the schedule length is longer than the values in **Experiment 1** in all cases, because the fast Fourier transform application has a better parallelism degrees than Gaussian elimination application. Similar to **Experiment 1**, the results indicate that ESECC still gives a better performance than MSLECC. The actual energy consumption values using HEFT clearly exceed the given energy consumption constraint.

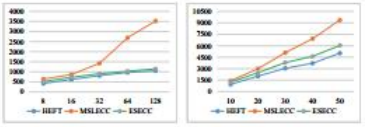
TABLE VII: Scheduling results of Gaussian elimination applications with ρ = 32(i.e., $|N|$ = 527) for varying energy consumption constraints

$E_{\text{given}}(G)$	HEFT		MSLECC		ESECC	
	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
9827.09	11667.62	3258	9827.09	4382.21	9826.80	3341.65
8423.22	11667.62	3258	8423.22	4965.82	8411.86	3679.22
7370.32	11667.62	3258	7370.32	5134.26	7368.53	3798.29
6551.40	11667.62	3258	6551.40	5249.42	6551.01	3803.11
5896.26	11667.62	3258	5896.26	5418.86	5896.15	3865.41

C. Varying number of tasks

This section we consider the parallel applications for varying scales, two experiments are performed. The details are described as follow:

Experiment 3. This experiment is designed to compare the actual energy consumption and final schedule length of the fast Fourier transform applications for varying scales. The application size is varying from $\rho = 8$ (Small scale) to $\rho = 128$ (Large scale), and the energy consumption constraint is set to $(E_{\min}(G) + E_{\max}(G))/10$.



(a) fast Fourier transform application (b) Gaussian elimination application

Fig. 5: Final schedule length for varying scales

Fig. 5(a) shows the final schedule length values of the fast Fourier transform applications for varying number of tasks by using the three algorithms. However, the actual energy consumption obtained by using HEFT cannot meet the constraint at each scale test in this paper. Meanwhile, using MSLECC and ESECC can always satisfy the energy consumption constraint, but the schedule length of MSLECC became more pessimism as the scale increased. As shown in Table VIII, when $\rho = 128$(i.e., $|N|$=1151) , the schedule length is 1145.83 using ESECC, and the value obtained by HEFT and MSLECC is 3524 and 1049 respectively. ESECC can reduce the schedule length by 67.5% than MSLECC, meanwhile, it is almost the same as HEFT while the actual energy consumption is merely 60.7% of the HEFT. The results shows that the proposed ESECC has a better performance than MSLECC, and it is highly effective while the scale of applications is very large.

TABLE VIII: Scheduling results of fast Fourier transform applications with energy consumption constraints for varying number of tasks

ρ	$E_{\text{given}}(G)$	HEFT		MSLECC		ESECC	
		$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
8	457.90	916.82	419	457.90	637.53	457.64	511.66
16	1101.10	1852.55	617	1101.10	862.26	1099.31	727.17
32	2643.70	5644.86	811	2643.70	1420.88	2641.30	903.41
64	6008.18	11966.81	965	6008.18	2690.00	6007.41	1026.06
128	12985.64	21388.55	1049	12985.64	3524.00	12985.07	1145.83

Experiment 4. This experiment is designed to compare the actual energy consumption and final schedule length of the Gaussian elimination applications for varying scales. The application size ρ is varying from 10 to 50 with 10 increment. The energy consumption constraint is set to $(E_{\min}(G) + E_{\max}(G))/10$.

Fig. 5(b) shows the final schedule length value of Gaussian elimination application for varying number of tasks by using the three algorithms. The detail values are shown in Table IX. We can see the schedule length is increased gradually with the scale of applications. Similar to the former experiments, the schedule length generated by ESECC is still shorter than MSLECC. The results of **Experiment 3** and **Experiment 4** indicate that the ESECC algorithm can apply to whether small-scale or large-scale applications, and also demonstrate its effectiveness in both high parallelism degrees and low parallelism degrees applications.

TABLE IX: Scheduling results of Gaussian elimination applications with energy consumption constraints for varying number of tasks

ρ	$E_{\text{given}}(G)$	HEFT		MSLECC		ESECC	
		$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
10	645.38	1323.57	945	645.38	1410.51	645.28	1241.31
20	2463.44	4372.55	2023	2463.44	2990.91	2463.26	2470.18
30	5444.30	10456.16	3050	5444.30	5088.47	5444.24	3791.76
40	9565.54	19503.5	3711	9565.54	6919.90	9564.14	4614.21
50	14899.41	31796.78	5027	14899.41	9370.67	14899.18	6040.58

VI. Conclusion

An efficient method of schedule length minimization for energy consumption constrained parallel applications on heterogeneous parallel distributed systems is provided in this study. First, the proposed algorithm can always meet the need of energy consumption constraint with the verified by using proof and experiments. Second, ESECC demonstrates more effective schedule length minimization than the state-of-the-art algorithms with low time complexity. Our algorithm effectively facilitates a part of energy-aware design for parallel applications on heterogeneous computing systems.

References

[1] Sharifi S, Coskun A K, Rosing T S. Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor SoCs[C]//Proceedings of the 2010 Asia and South Pacific Design Automation Conference. IEEE Press, 2010: 873-878.

[2] Li K. Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers[J]. IEEE Transactions on Computers, 2012, 61(12): 1668-1681.

[3] Weiser M, Welch B, Demers A, et al. Scheduling for reduced CPU energy[M]//Mobile Computing. Springer US, 1994: 449-471.

[4] Barnett J A. Dynamic task-level voltage scheduling optimizations[J]. IEEE Transactions on Computers, 2005, 54(5): 508-520.

[5] Zhong X, Xu C Z. Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee[J]. IEEE Transactions on computers, 2007, 56(3).

[6] Niu L, Quan G. Energy-aware scheduling for practical mode real-time systems with QoS guarantee[C]//Computer Science and Information Engineering, 2009 WRI World Congress on. IEEE, 2009, 3: 428-432.

[7] Lee Y C, Zomaya A Y. Energy conscious scheduling for distributed computing systems under different operating conditions[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(8): 1374-1381.

[8] Li K. Power and performance management for parallel computations in clouds and data centers[J]. Journal of Computer and System Sciences, 2016, 82(2): 174-190.

[9] Huang Q, Su S, Li J, et al. Enhanced energy-efficient scheduling for parallel applications in cloud[C]//Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012: 781-786.

[10] Tang Z, Qi L, Cheng Z, et al. An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment[J]. Journal of Grid Computing, 2016, 14(1): 55-74.

[11] Rusu C, Melhem R, Moss D. Maximizing rewards for real-time applications with energy constraints[J]. ACM Transactions on Embedded Computing Systems (TECS), 2003, 2(4): 537-559.

[12] Xiao X, Xie G, Li R, et al. Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems[C]//Trustcom/BigDataSE/I SPA, 2016 IEEE. IEEE, 2016: 1471-1476.

[13] Bunde D P. Power-aware scheduling for makespan and flow[J]. Journal of Scheduling, 2009, 12(5): 489-500.

[14] Rizvandi N B, Taheri J, Zomaya A Y. Some observations on optimal frequency selection in DVFS-based energy consumption minimization[J]. Journal of Parallel and Distributed Computing, 2011, 71(8): 1154-1164.

[15] Xie G, Jiang J, Liu Y, et al. Minimizing Energy Consumption of Real-Time Parallel Applications using Downward and Upward Approaches on Heterogeneous Systems[J]. IEEE Transactions on Industrial Informatics, 2017.

[16] Topcuoglu H, Hariri S, Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE transactions on parallel and distributed systems, 2002, 13(3): 260-274.

[17] Zhu D, Aydin H. Reliability-aware energy management for periodic real-time tasks[J]. IEEE Transactions on Computers, 2009, 58(10): 1382-1397.

[18] Zhao B, Aydin H, Zhu D. On maximizing reliability of real-time embedded applications under hard energy constraint[J]. IEEE Transactions on Industrial Informatics, 2010, 6(3): 316-328.

[19] Zhao B, Aydin H, Zhu D. Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2013, 18(2): 23.

接下来

- 继续修改完善论文。

THANKS