# Minimizing Schedule Length of Energy Consumption Constrained Parallel Applications on Heterogeneous Distributed Systems

Xiongren Xiao[1,2], Guoqi Xie[1,2,*], Renfa Li[1,2], and Keqin Li[1,3]
[1]College of Computer Science and Electronic Engineering, Hunan University, China
[2]Key Laboratory for Embedded and Network Computing of Hunan Province, China
[3]Department of Computer Science, State University of New York, USA
{xxr@hnu.edu.cn, xgqman@hnu.edu.cn, lirenfa@hnu.edu.cn, lik@newpaltz.edu}
[*]Corresponding author

***Abstract*—Energy consumption is one of the primary design constraints in heterogeneous parallel and distributed systems ranging from small embedded devices to large-scale data centers. The problem of minimizing schedule length of an energy consumption constrained parallel application has been studied recently in homogeneous systems with shared memory. To adapt the heterogeneity and distribution of high-performance computing systems, this study solves the problem of minimizing schedule length of an energy consumption constrained parallel application on heterogeneous distributed systems based on dynamic voltage and frequency scaling (DVFS) energy-efficient design technique. Such problem is decomposed into two sub-problems in this study, namely, satisfying energy consumption constraint and minimizing schedule length. The first sub-problem is solved by transferring the energy consumption constraint of the application to that of each task, and the second sub-problem is solved by heuristically scheduling each task with low time complexity. Experiments with Fast Fourier transform parallel applications show that not only the actual energy consumptions always do not exceed and are close to given energy consumption constraints, but also the minimum schedule lengths are generated by using the proposed algorithm.**

***Keywords*—energy consumption, heterogeneous systems, parallel applications, schedule length**

## I. Introduction

### A. Background

Recent trends in the microprocessor industry have important for the design of high-performance computing systems. By increasing number of heterogeneous processors and cores, it is possible to improve the performance while keeping the energy consumption at the bay. This trend has reached the deployment stage in heterogeneous parallel and distributed systems ranging from small embedded devices to large-scale data centers. It is expected that the number of heterogeneous processor and cores in these systems increases dramatically in the near future. For such systems, energy consumption is one of the primary design constraints. The popular energy consumption optimization technique dynamic voltage and frequency scaling (DVFS) achieves energy-efficient optimization by simultaneously scaling down processor's supply voltage and frequency while tasks are running to explore the tradeoff between energy consumption and execution time [1].

### B. Related works

DVFS-based energy-efficient design technique was first introduced in [2]. In [3], the authors studied the energy-aware task scheduling of independent sequential tasks on homogeneous multi-processors as combinatorial optimization problems. In [4], the authors simultaneously addressed three constraints (i.e., energy, deadline, and reward) for both homogeneous and heterogeneous systems. In [5], the authors studied the problem of scheduling a collection of independent tasks with deadlines and energy consumption constraints on heterogeneous systems.

The limitation of the aforementioned works is quite restricted to independent tasks. However, parallel applications (e.g., Fast Fourier transform applications [6]) with precedence constrained tasks are widely used in high-performance heterogeneous distributed computing systems, In [7], the authors considered energy-aware duplication scheduling algorithms for a parallel application on homogeneous systems, and in [8], the authors presented energy-conscious scheduling (ECS) to implement joint minimization of energy consumption and schedule length of a parallel application on heterogeneous distributed systems. The problem of minimizing schedule length of an energy consumption constrained application with precedence constrained sequential tasks [1] and precedence constrained parallel tasks (i.e., a parallel application) [9] were solved, respectively. These two works were merely interested in homogeneous systems with shared memory and it cannot be applied to heterogeneous distributed systems with communication time between any two tasks. This study aims to implement the objective of minimizing schedule length of an energy consumption constrained parallel application on heterogeneous distributed systems.

## II. Models

### A. Application model

Let $U = \{u_1, u_2, ..., u_{|U|}\}$ represent a set of heterogeneous processors, where $|U|$ represents the size of set $U$. Note that for any set $X$, this study uses $|X|$ to denote its size. A parallel application running on processors is represented by a directed acyclic graph (DAG) $G$=($N$, $M$, $C$, $W$) [6], [8],

2324-9013/16 $31.00 © 2016 IEEE
DOI 10.1109/TrustCom/BigDataSE/ISPA.2016.228

[10]. $N$ represents a set of nodes in $G$, and each node $n_i \in N$ represents a task with different execution times on different processors. $M$ 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, $c_{i,j} \in C$ represents communication time of $m_{i,j}$ if $n_i$ and $n_j$ are not assigned to the same processor. $pred(n_i)$ represents the set of the immediate predecessor tasks of $n_i$. $succ(n_i)$ represents the set of the immediate successor tasks of $n_i$. The task which has no predecessor task is denoted as $n_{\text{entry}}$; and the task which has no successor task is denoted as $n_{\text{exit}}$. $W$ is a $|N|\times|U|$ matrix where $w_{i,k}$ denotes the execution time of $n_i$ runs on $u_k$ with the maximum frequency.

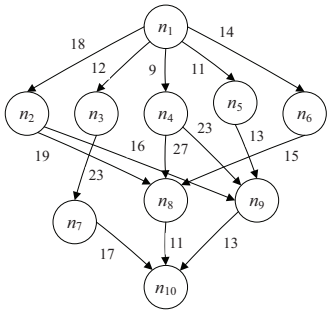


Fig. 1: A motivating example of a DAG-based parallel application with ten tasks.

Fig. 1 shows a motivating example of a DAG-based parallel application. Table 1 is a matrix of execution times with the maximum frequency in Fig. 1. The example shows ten tasks executed on three processors $\{u_1, u_2, u_3\}$. The weight 14 of $n_1$ and $u_1$ in Table 1 represents the execution time denoted by $w_{1,1}$=14. We can see that the same task has different execution times on different processors due to the heterogeneity of the processors. The weight 18 of edge (Fig. 1) between $n_1$ and $n_2$ represents the communication time denoted as $c_{1,2}$ if $n_1$ and $n_2$ are not assigned to the same processor.

TABLE 1: Execution time of tasks on different processors with the maximum frequency of the parallel application in Fig. 1.

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

### *B. Power and energy model*

As the almost linear relationship between the voltage and frequency, DVFS scales down the voltage alongside the frequency to save energy. Similar to [11], [12], we use the term frequency change to stand for changing the voltage and frequency simultaneously. Considering a DVFS-capable system, we also adopt the system-level power model widely used in [11], [12], where the power consumption at frequency $f$ is given by

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

$P_s$ represents the static power and can be removed only by powering off the whole system. $P_{\text{ind}}$ represents frequency-independent dynamic power and can be removed by putting the system into the sleep state. $P_{\text{d}}$ represents frequency-dependent dynamic power, and depends on frequencies. $h$ represents system states and indicates whether dynamic powers are currently consumed in the system. When the system is active, $h$ = 1; otherwise, $h$ = 0. $C_{\text{ef}}$ represents effective switching capacitance and $m$ represents the dynamic power exponent and is no smaller than 2. Both $C_{\text{ef}}$ and $m$ are processor-dependent constants.

Note that there exists an excessive overhead associated with turning on/off a system, $P_{\text{s}}$ is always consumed and not manageable [11], [12]. Similar to the above works, this study concentrates on managing the dynamic power (i.e., $P_{\text{ind}}$ and $P_{\text{d}}$). Because of the $P_{\text{ind}}$, less $P_{\text{d}}$ does not result less energy consumption. That is, a minimum energy-efficient frequency $f_{\text{ee}}$ exists [11], [12] and it is denoted as

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

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

As the number of processors is $|U|$ in the system and these processors are completely heterogeneous, each processor should has individual power parameters; Here, we define frequency-independent dynamic power set $\{P_{1,\text{ind}}, P_{2,\text{ind}}, ..., P_{|U|,\text{ind}}\}$, frequency-dependent dynamic power set $\{P_{1,\text{d}}, P_{2,\text{d}}, ..., P_{|U|,\text{d}}\}$, effective switching capacitance set $\{C_{1,\text{ef}}, C_{2,\text{ef}}, ..., C_{|U|,\text{ef}}\}$, dynamic power exponent set $\{m_1, m_2, ..., m_{|U|}\}$, minimum energy-efficient frequency set $\{f_{1,\text{ee}}, f_{2,\text{ee}}, ..., f_{|U|,\text{ee}}\}$, and actual effective frequency set

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

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

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

where

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

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

## III. PRELIMINARIES

### A. Energy consumption constraint

As the execution time of each task on each processor is known, we can get the minimum and maximum energy consumption denoted by $E_{\min}(n_i)$ and $E_{\max}(n_i)$, respectively, by traversing all the processors. $E_{\min}(n_i)$ and $E_{\max}(n_i)$ are obtained by executing the task with the maximum and minimum frequencies, respectively. Both of them are calculated by

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

and

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

respectively.

As the energy consumption of the application $G$ is the sum of that of each task, we can obtain that the minimum and maximum energy consumption of $G$ are

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

and

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

respectively.

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

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

### B. Problem description

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

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

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

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

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

### C. Task prioritizing

We first need to determine the task assignment order before assigning tasks to processors. Similar to [6], [8], we employ the upward rank value ($rank_{\text{u}}$) of a task given by Eq. (10) as the common task priority standard. All the tasks are ordered according to the decreasing order of $rank_{\text{u}}$.

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

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

## IV. SCHEDULING POLICY

The problem that minimizing the schedule length of an energy consumption constrained parallel application on heterogeneous distributed systems is decomposed to two subproblems, namely, satisfying energy consumption constraint and minimizing schedule length. We first solve these two sub-problems separately, and then present the algorithm by integrating the two sub-problems.

### A. Satisfying energy consumption constraint

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

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

### B. Minimizing schedule length

HEFT is the well-known precedence-constrained task scheduling based on the DAG model to reduce schedule length to a minimum combined with low complexity and high performance in heterogeneous systems [6]. Besides the task prioritizing based on upward rank value, task assignment based on earliest finish time (EFT) was also presented. As the original EFT does not consider the frequency adjustment, new EFT should be represented.

Let $EST(n_i, u_k, f_{k,h})$ and $EFT(n_i, u_k, f_{k,h})$ represent the earliest start time (EST) and EFT, respectively, of the

task $n_i$ on the processor $u_k$ with the frequency $f_{k,h}$. The aforementioned attributes are calculated as

$$\begin{cases} EST(n_{\text{entry}},u_k,f_{k,h})=0 \\ EST(n_i,u_k,f_{k,h})=\max(avail[k], \max\limits_{n_x\in pred(n_i)}\{AFT(n_x)+c'_{x,i}\} \end{cases}, \tag{12}$$

and

$$EFT(n_i,u_k,f_{k,h}) = EST(n_i,u_k,f_{k,h}) + w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{13}$$

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

### C. Scheduling algorithm

We first give the energy consumption constraint of each task before we propose the algorithm. According to Eq. (11), we have

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

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

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

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

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

Hence, when assigning the task $n_{seq(j)}$, we can directly consider the energy consumption constraint $E_{\text{given}}(n_{seq(j)})$ of $n_{seq(j)}$ and do not have to be concerned about the energy consumption constraint of the application $G$. In this way, a low time complexity heuristic algorithm can be achieved. As the maximum energy consumption constraint of $n_{seq(j)}$ is $E_{\max}(n_i)$, $E_{\text{given}}(n_{seq(j)})$ should be required to satisfy the following constraint:

$$E(n_{seq(j)},u_k,f_{k,h}) \leqslant \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}. \tag{15}$$

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

**Algorithm 1** The MSLECC Algorithm

```
1: Sort the tasks in a list downward_task_list by descending order of rank_u values.
2: while (there are tasks in downward_task_list) do
3:     n_i = downward_task_list.out();
4:     Calculate E_min(n_i) and E_max(n_i) using Eqs. (4) and (5), respectively;
5:     Calculate E_given(n_i) using Eq. (14);
6:     var pr(i) = NULL, f_{pr(i),hz(i)} = NULL, AFT(n_i) = ∞, E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = 0;
7:     for (each processor u_k ∈ U) do
8:         for (each frequency f_{k,h} in the scope of [f_{k,low}, f_{k,max}]) do
9:             Calculate E(n_i, u_k, f_{k,h}) using Eq. (2);
10:            if (E(n_i, u_k, f_{k,h}) > min{E_given(n_i), E_max(n_i)}) then
11:                continue; // skip the processor and frequency that do not satisfy the energy consumption constraint of n_i
12:            end if
13:            Calculate EFT(n_i, u_k, f_{k,h}) using Eq. (13);
14:            if (EFT(n_i, u_k, f_{k,h}) < AFT(n_i)) then
15:                pr(i) = k;
16:                f_{pr(i),hz(i)} = f_{k,h};
17:                E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = E(n_i, u_k, f_{k,h});
18:                AFT(n_i) = EFT(n_i, u_k, f_{k,h}); // select the processor and frequency with the minimum EFT
19:            end if
20:        end for
21:    end for
22: end while
23: Calculate the actual energy consumption E(G) using Eq. (9);
24: Calculate SL(G) = AFT(n_exit);
```

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

1) In Line 6, we initialize $AFT(n_i) = \infty$ and $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = 0$.

2) In Lines 7-21, we traverse all processors and frequencies and select the processor with the minimum EFT for each task under satisfying the condition of $E(n_i,u_k,f_{k,h}) \leqslant \min\{E_{\text{given}}(n_i), E_{\max}(n_i)\}$.

3) In Lines 23 and 24, calculate the actual energy consumption $E(G)$ and final schedule length $SL(G)$, respectively.

4) As MSLECC is a heuristic algorithm, it has a low time complexity of $O(|N|^2 \times |U| \times |F|)$, where $F$ represents the maximum number of discrete frequencies from the lowest to the maximum actual effective frequencies. In other words, MSLECC implements low time complexity and high-performance scheduling for energy consumption constrained parallel applications.

### D. Example of the MSLECC algorithm

We assume that the power parameters for all processors are known and shown in Table 2, where the maximum frequency $f_{k,\max}$ for each processor is 1 and the frequency precision is set as 0.01. We can obtain the minimum energy-efficient frequency $f_{k,\text{ee}}$ (considered as the $f_{k,\text{low}}$ in this example) for each processor and dynamic power of $p_{k,h}$ according to Eqs. (1) and (3), respectively.

We can calculate that the minimum and maximum reliability values are $E_{\min}(G) = 20.31$ and $E_{\max}(G) = 161.99$

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

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

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

TABLE 3: Task assignment of the application in Fig. 1 using MSLECC.

| $n_i$ | $E_{given}(n_i)$ | $u_{pr(i)}$ | $f_{pr(i),hz(i)}$ | $E(n_i, pr(i), f_{pr(i),hz(i)})$ | $AST(n_i)$ | $AFT(n_i)$ |
|---|---|---|---|---|---|---|
| $n_1$ | 13.44 | $u_3$ | 1.0 | 9.63 | 0 | 12 |
| $n_3$ | 20.33 | $u_3$ | 1.0 | 20.33 | 9 | 28 |
| $n_4$ | 18.19 | $u_2$ | 1.0 | 6.72 | 18 | 26 |
| $n_2$ | 19.26 | $u_1$ | 1.0 | 10.79 | 27 | 40 |
| $n_5$ | 10.92 | $u_3$ | 1.0 | 10.7 | 28 | 38 |
| $n_6$ | 13.44 | $u_2$ | 1.0 | 13.44 | 26 | 42 |
| $n_9$ | 5.4385 | $u_2$ | 0.61 | 5.3606 | 56 | 75.67 |
| $n_7$ | 1.3188 | $u_1$ | 0.33 | 1.3177 | 51 | 72.2121 |
| $n_8$ | 0.8874 | $u_1$ | 0.26 | 0.8863 | 72.2121 | 91.4429 |
| $n_{10}$ | 1.8204 | $u_2$ | 0.26 | 1.8193 | 102.4429 | 129.3660 |
| $E(G) = 80.98 \leqslant E_{\text{given}}(G) = 80.9939$, $SL(G) = AFT(n_{10})$=129.3660 | | | | | | |

Fig. 2 also shows the scheduling of the parallel application $G$ in Fig. 1 using MSLECC, where the schedule length is 121.84. Note that the arrows in Fig. 2 represent generated communication times between tasks.

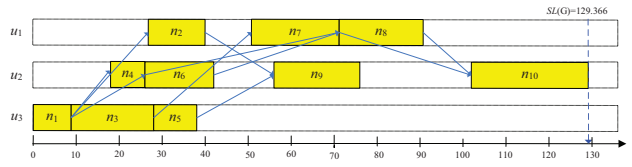


Fig. 2: Scheduling of the application in Fig. 1 using MSLECC.

## V. Experiments

The performance metrics selected for comparison are the actual energy consumption $E(G)$ (Eq. (9)) and the final schedule length $SL(G)$ of application. The compared algorithms with our proposed MSLECC are HEFT [6] and ECS [8] because all of them have the same application model. Processor and application parameters are below: 10 ms $\leqslant w_{i,k} \leqslant$ 100 ms, 10 ms $\leqslant c_{i,j} \leqslant$ 100 ms, $0.03 \leqslant P_{k,\text{ind}} \leqslant 0.07$, $0.8 \leqslant C_{k,\text{ef}} \leqslant 1.2$, $2.5 \leqslant m_k \leqslant 3.0$, and $f_{k,\max} = 1$ GHz. All frequencies are discrete, and the precision is 0.01 GHz. All parallel applications will be executed in a simulated heterogeneous multi-processors platform with 64 processors. To verify the effectiveness and reality, we use Fast Fourier transform applications to observe the results.

A new parameter $\rho$ is used as the size of the Fast Fourier transform application, and the total number of tasks is [6] $|N| = (2\times\rho-1)+\rho\times\log_2\rho$, where $\rho = 2^y$ for some integer $y$. Fig. 3 shows an example of the Fast Fourier transform parallel application with $\rho$=8. Note that $\rho$ exit tasks exist in the Fast Fourier transform application with the size $\rho$. To adapt the application model of this study, we just add a virtual exit task and the last $\rho$ tasks are set as the immediate predecessor tasks of the virtual task.

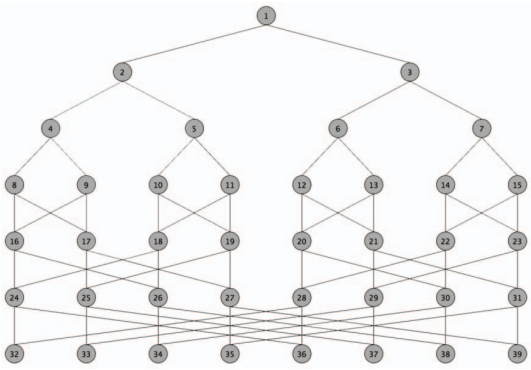

Fig. 3: Example of the Fast Fourier transform parallel application with $\rho$=8.

**Experiment 1**. This experiment is conducted to compare the actual energy consumptions and final schedule length of Fast Fourier transform parallel applications for varying energy consumption constraints. We limit the size of the application as $\rho = 32$ (i.e., $|N| = 233$). $E_{\text{given}}(G)$ is changed from $(E_{\min}(G) + E_{\max}(G))/10$ to $(E_{\min}(G) + E_{\max}(G))/6$.

TABLE 4: Actual energy consumptions (kWs) and final schedule length (ms) of Fast Fourier transform parallel applications with $\rho = 32$ for varying energy consumption constraints.

| $E_{\min}(G)$ | $E_{\max}(G)$ | $E_{\text{given}}(G)$ | HEFT [6] | | ECS [8] | | MSLECC | |
|---|---|---|---|---|---|---|---|---|
| | | | $E(G)$ | $SL(G)$ | $E(G)$ | $SL(G)$ | $E(G)$ | $SL(G)$ |
| 654.63 | 26328.44 | 2698.30 | 8809.7 | 811 | 5913.40 | 1055.38 | 2698.30 | 1386.57 |
| 618.91 | 26304.68 | 2991.51 | 8392.5 | 893 | 5928.01 | 927.12 | 2991.50 | 1082.6 |
| 622.91 | 25829.49 | 3306.55 | 8057.6 | 797 | 5856.44 | 894.25 | 3306.53 | 1051.56 |
| 649.26 | 26372.31 | 3860.22 | 8949.7 | 916 | 6091.70 | 1092.67 | 3860.21 | 1311.72 |
| 629.15 | 26289.34 | 4486.41 | 8852.2 | 847 | 6049.71 | 867.21 | 4486.41 | 887.76 |

We can see from Table 4 that the actual energy consumptions of applications using both HEFT and ECS cannot satisfy individual energy consumption constraints in all cases. Such results verify that ECS is not designed for satisfying the energy consumption constraints of applications in practice. On the contrary, MSLECC always can satisfy the energy consumption constraints and the actual energy consumptions are increasingly close to the energy consumption constraints. For example, when $E_{\text{given}}(G) = 4486.41$, the energy consumptions using HEFT and ECS are 8852.2 and 6049.71 kWs, respectively, whereas that using MSLECC is 4486.41 kWs, which is much close to 4486.41 kWs. We can also see that the schedule lengths using MSLECC have been effectively controlled in acceptable scopes under satisfying the energy consumption constraints although the schedule lengths using MSLECC are

slightly longer than that using MSLECC and ECS in this experiment.

**Experiment 2**. To observe the performance in different scales of applications, this experiment is conducted to compare the actual energy consumptions and final schedule lengths of Fast Fourier transform parallel applications for varying number of tasks. We limit $E_{\text{given}}(G)$ as $E_{\text{given}}(G) = (E_{\min}(G) + E_{\max}(G))/6$. $\rho$ is changed from 8 to 128, namely, the number of tasks are changed from 33 (small scale) to 1151 (large scale).

TABLE 5: Actual energy consumptions (kWs) and final schedule length (ms) of Fast Fourier transform parallel applications with for varying number of tasks.

| $\rho$ | $\|N\|$ | $E_{\min}(G)$ | $E_{\max}(G)$ | $E_{\text{given}}(G)$ | HEFT [6] | | ECS [8] | | MSLECC | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | $E(G)$ | $SL(G)$ | $E(G)$ | $SL(G)$ | $E(G)$ | $SL(G)$ |
| 8 | 39 | 112.62 | 4647.52 | 793.353 | 1425.52 | 459.9 | 943.11 | 537.79 | 793.33 | 553.84 |
| 16 | 95 | 264.21 | 11324.02 | 1931.37 | 3591.45 | 611.47 | 2413.73 | 708.71 | 1931.36 | 750.92 |
| 32 | 233 | 630.03 | 26226.91 | 4476.15 | 8463.99 | 859.7 | 5975.67 | 925.11 | 4476.14 | 979.77 |
| 64 | 511 | 1442.85 | 59923.89 | 10227.79 | 18685.90 | 1450.6 | 14241.24 | 1187.3 | 10213.94 | 1162.65 |
| 128 | 1151 | 3091.14 | 132705.98 | 22632.85 | 38737.37 | 1950.8 | 29517.43 | 1546.5 | 22598.84 | 1329.0 |

We can see from Table 5 that the energy consumption constraints and actual energy consumptions are increased gradually with the increase of the number of tasks. However, the actual energy consumptions of applications using HEFT and ECS still cannot satisfy their energy consumption constraints in different scales. With the increasing number of tasks, the differences between $E_{\text{given}}(G)$ and $E(G)$ increase greatly. On the contrary, MSLECC always can satisfy the energy consumption constraints and the actual energy consumptions are still close to the energy consumption constraints. For example, when $|N| = 1151$, the energy consumption constraint is $E_{\text{given}}(G)$ =22632.85 kWs, but the actual energy consumptions using HEFT and ECS are 38737.3 and 29517.43 kWs, respectively, which obviously exceed given energy consumption constraint. Fortunately, the actual energy consumptions using MSLECC is 22598.84 kWs, which is close to the given energy consumption constraint, and the difference is merely 34.01 kWs.

In addition to satisfying the energy consumption constraints, an exciting phenomenon is that MSLECC can also generate shorter schedule lengths than HEFT and ECS in large-scale parallel applications (e.g., $|N| = 511$ and $|N| = 1151$). For example, when $|N| = 1151$, the schedule length using MSLECC is 1329 kWs, which is much less than 1950 and 1546.5 kWs that using HEFT and ECS. Such results indicate that: 1) MSLECC is very suitable for minimizing schedule length of energy consumption constrained parallel applications and 2) energy consumption optimization is extremely desirable and useful for large-scale parallel applications.

## VI. Conclusions

We have developed an effective and low time complexity schedule length minimization algorithm MSLECC for an energy consumption constrained parallel application on heterogeneous distributed systems based on DVFS energy-efficient design technique. First, our algorithm can always satisfy the energy consumption constraint, and its correctness is verified using proof and experiments. Second, our MSLECC algorithm implements effective and low time complexity task scheduling to minimize the schedule length. We believe that our MSLECC algorithm could effectively improve a part of energy-aware design for parallel applications in heterogeneous distributed environments during the design phases.

## Acknowledgment

This work was partially supported by the National High-Tech Research and Development Plan of China under Grant No. 2012AA01A301-01, the Key Program of National Natural Science Foundation of China under Grant No. 61432005, and the National Natural Science Foundation of China under Grant Nos. 61173036,61370095, 61502162, and 61370097, and the China Postdoctoral Science Foundation under Grant No. 2016M592422.

## References

[1] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *Computers, IEEE Transactions on*, vol. 61, no. 12, pp. 1668–1681, 2012.

[2] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced cpu energy," in *Mobile Computing*. Springer, 1996, pp. 449–471.

[3] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 1484–1497, 2008.

[4] C. Rusu, R. Melhem, and D. Mossé, "Maximizing rewards for real-time applications with energy constraints," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 2, no. 4, pp. 537–559, 2003.

[5] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 11, pp. 2867–2876, 2014.

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

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

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

[9] K. Li, "Power and performance management for parallel computations in clouds and data centers," *Journal of Computer and System Sciences*, vol. 82, no. 2, pp. 174–190, 2016.

[10] G. Xie, R. Li, and K. Li, "Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems," *Journal of Parallel and Distributed Computing*, vol. 83, pp. 1–12, 2015.

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

[12] B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," *Acm Transactions on Design Automation of Electronic Systems*, vol. 18, no. 2, pp. 99–109, 2013.