Future Generation Computer Systems 105 (2020) 916–931

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Energy management for multiple real-time workflows on cyber–physical cloud systems

Guoqi Xie [a,b,*], Gang Zeng [c], Junqiang Jiang [a], Chunnian Fan [d], Renfa Li [a,b], Keqin Li [a,e]

[a] College of Computer Science and Electronic Engineering, Hunan University, China
[b] Key Laboratory for Embedded and Network Computing of Hunan Province, China
[c] Graduate School of Engineering, Nagoya University, Japan
[d] Nanjing University of Information Science and Technology, China
[e] Department of Computer Science, State University of New York, New Paltz, NY, 12651, USA

HIGHLIGHTS

- Maximizing the number of workflows that are completed within their deadlines by proposing a deadline-driven processor merging for multiple workflows (DPMMW) algorithm.
- Minimizing the energy consumption of the workflows that are completed within their deadlines by proposing a global energy saving for multiple workflows (GESMW) algorithm.
- Experimental results validate that the combined DPMMW&GESMW algorithm can reduce deadline miss ratio (DMR) and save satisfactory energy over the existing methods.

ARTICLE INFO

Article history:
Received 28 November 2016
Received in revised form 15 May 2017
Accepted 22 May 2017
Available online 26 May 2017

Keywords:
Cyber–physical cloud systems (CPCS)
Deadline miss ratio (DMR)
Global energy saving (GES)
Multiple workflows
Real-time constraint

ABSTRACT

Cyber–physical cloud systems (CPCS) are extensions of cyber–physical systems (CPS) that expand the cyber-part and distribute it on-device and in-cloud. CPCS are considered large-scale heterogeneous distributed cloud computing systems that support execution of multiple workflows. This study aims to reduce the energy consumption of multiple real-time workflows on CPCS and it contains two objectives: (1) maximizing the number of workflows that are completed within their deadlines; (2) minimizing the energy consumption of the workflows that are completed within their deadlines. The former is solved by proposing a deadline-driven processor merging for multiple workflows (DPMMW) algorithm, whereas the latter is solved by proposing a global energy saving for multiple workflows (GESMW) algorithm to minimize the total energy consumption. Experimental results validate that the combined DPMMW&GESMW algorithm can reduce deadline miss ratio (DMR) and save as much as possible energy over existing methods.

1. Introduction

1.1. Background

Cyber–physical systems (CPS), which were first coined at the National Science Foundation (NSF) in the United States (US) in 2006, are engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components [1]. Advances in CPS will enable capability, adaptability, scalability, resiliency, safety, security, and usability that will further enhance the existing simple embedded systems [1]. Considerable progress has been made in developing CPS technology over the past five years in many sectors, such as vehicular CPS (VCPS) [2], automotive CPS (ACPS) [3], medical CPS (MCPS) [4], and cyber–physical social systems (CPSS) [5,6], among others.

The following are the two key parts integrated in balance of CPS: (1) the physical part, including sensors and actuator constellations, and interacts with the physical environments, such as lighting, temperature, water, and fertilizers; (2) the cyber part, including the computation and communication resources, and manages and enhances the hardware capabilities and its interaction with the

* Corresponding author at: College of Computer Science and Electronic Engineering, Hunan University, China.
E-mail addresses: xgqman@hnu.edu.cn (G. Xie), sogo@ertl.jp (G. Zeng), jjq@hnu.edu.cn (J. Jiang), fcn@nuist.edu.cn (C. Fan), lirenfa@hnu.edu.cn (R. Li), lik@newpaltz.edu (K. Li).

http://dx.doi.org/10.1016/j.future.2017.05.033
0167-739X/

cyber-world [7]. The rapid advances in computational power coupled with the prevalence of the cloud and its services benefits, enables us to expand the cyber part and distribute it on-device and in-cloud [7,8]. That is, a new generation of CPS called cyber–physical cloud systems (CPCS) or cloud-based CPS emerge [7,8]. Integrating CPS into a cloud computing infrastructure forms CPCS that improves interaction among cyber–physical devices and also enables large-scale data storage, analysis, and service [7,8]. CPCS is an extension of cloud data centers by including the physical part into the infrastructure.

Given the continuous replacement of old and slow machines with new and fast ones, CPCS are believed to become heterogeneous. In X as a service (XaaS) clouds, resources as services (e.g., infrastructure, platform, and software) are sold to workflows, such as scientific and big data analysis workflows [9–13]. As heterogeneous systems continue to be scaled up, workflows with precedence-constrained tasks, such as fast Fourier transform and Gaussian elimination [14–16]. A task graph is a workflow model for the description of distributed end-to-end computations in CPCS. Given that a workflow is released by receiving collected data from the sensor and is completed by sending the performing action to the actuator in CPCS, the task graph is restricted to be directed and acyclic, and is then represented by a directed acyclic graph (DAG) [14–16]. Such a workflow is called DAG-based workflow. Furthermore, CPCS are large-scale systems that accommodate the execution of multiple workflows, which represent multiple DAGs in heterogeneous distributed systems [17–23]. Therefore, CPCS are considered as large-scale heterogeneous distributed cloud computing systems.

1.2. Motivations

Energy sustainability is an important part of energy provision and management policies because it has a direct environmental impact. The terms "sustainability" and "energy" elicited considerable concern in NSF 16-549 [1]. Therefore, CPCS are expected to play a major role in the development of next-generation smart energy systems and data centers. Innovative computational methodologies such as green and energy efficient CPCS design have become critical to enable the sustainable development of such systems. These technologies can be used to tackle the reduction of energy induced from the large scale data center computing infrastructures, and the improvement of computational efficiency in smart energy systems.

In addition to energy management, emergency response is an important real-time constraint in CPCS. For example, a CPCS workflow is released by receiving collected data from the sensor and is then completed by sending the performed action to the actuator. Many such workflows should be performed in individual deadlines. Cloud service providers and users are the two types of roles with conflicting requirements in CPCS. Minimizing the total energy consumption of a workflow is one of the most important concerns for cloud service providers; whereas, satisfying the real-time constraint of a workflow is one of the most important quality of service (QoS) requirements for users [21]. If the workflow cannot be finished in a given deadline, then the service-level agreement (SLA) is violated by resource providers and the workflow does not function correctly. However, CPCS cannot always satisfy the deadlines of all workflows. To the best of our knowledge, recent studies have been merely interested in reducing the energy consumption and stratifying real-time constraints for a single workflow in heterogeneous cloud environments [15,16], and no related works have been reported on multiple real-time workflows.

1.3. Our contributions

This study aims to reduce the energy consumption of multiple real-time workflows on CPCS and it contains two objectives: (1) maximizing the number of workflows that are completed within their deadlines; (2) minimizing the energy consumption of the workflows that are completed within their deadlines. The contributions of this study are summarized as follows.

(1) The first objective is solved by using a deadline-driven processor merging for multiple workflows (DPMMW) algorithm. This algorithm iteratively merges processors for each workflow using the well-studied heterogeneous earliest finish time (HEFT) algorithm until its deadlines cannot be met without involving energy saving, such that DPMMW can satisfy the deadlines of more workflows than the existing method.

(2) The second objective is solved by using a global energy saving for multiple workflows (GESMW) algorithm. This algorithm can reclaim slack times of all processors for each task by moving them to other processors using a global approach, such that GESMW is more energy-efficient than the existing method that reclaims slack times on the fixed processor for each task.

(3) We conduct experiments on various real workflows. Experimental results validate that the combined DPMMW&GESMW algorithm can reduce deadline miss ratio (DMR) and save as much as possible energy over the existing methods.

The rest of this paper is organized as follows. Section 2 reviews the related literature. Section 3 constructs a series of models for CPCS. Section 4 analyzes existing algorithms and then proposes DPMMW and GESMW algorithms. Section 5 validates the combined DPMMW&GESMW algorithm. Section 6 concludes this study.

2. Related works

The popular energy consumption optimization technique, namely, dynamic voltage and frequency scaling (DVFS), achieves energy-efficient optimization by simultaneously scaling down the supply voltage and frequency of a processor [24–30]. Given that this study focuses on energy management and real-time constraint of DAG-based workflows on CPCS, this section first reviews the related research as follows: (1) energy management for CPS; (2) energy management for single DAG-based workflow; and (3) resource management for multiple DAG-based workflows.

(1) Energy management for CPS. In [31], the authors studied energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers using a CPS approach. In [32,33], the authors presented a CPS approach to energy management in data centers based on DVFS. In [34], the authors discussed the research directions for energy-sustainable CPS. In [35], the authors discussed scheduling co-design of periodic tasks for CPS reliability and energy. Furthermore, cyber–physical energy systems (CPES) are defined as an independent term and are studied recently. In [36], the authors studied efficient utilization of renewable energy sources of gridable vehicles in CPES. In [37], the authors studied the simulation of CPES on challenges, tools, and methods. In [38], the authors focused on smart buildings using a CPES approach. However, these studies do not involve end-to-end computations for a CPCS workflow.

(2) Energy management for single workflow. The authors in [39] considered energy-aware duplication scheduling algorithms for a workflow on homogeneous systems, and the authors in [40] presented energy-conscious scheduling to implement joint minimization of energy consumption and execution time of a workflow on heterogeneous multiprocessor systems. However, these works fail to consider the real-time constraint of the workflow. Some studies solved the problem of energy consumption optimization of a real-time workflow using precedence-constrained

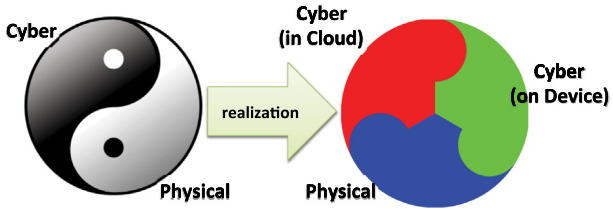


Fig. 1. Realization of CPS to CPCS [7].

sequential [28] and precedence-constrained parallel tasks [29,30]. However, these tasks are focused mostly on homogeneous multiprocessor systems with shared memory. The authors in [15] studied the problem of minimizing energy consumption of a real-time workflow on heterogeneous multiprocessor systems using energy-efficient scheduling (EES) algorithm by recycling slack time for each task on its fixed assigned processor. The authors in [16] solved the same problem using the DVFS-enabled energy-efficient workflow task scheduling (DEWTS). The DEWTS introduces a feature that can turn off relatively inefficient processors to reduce the static energy consumption and to realize EES-based slack time reclamation [15]. However, EES and DEWTS are merely interested in reducing the energy consumption for a single workflow.

(3) Resource management for multiple workflows. Two types exist in multiple DAG-based workflows, including static and dynamic scheduling. All workflows are released simultaneously in static scheduling [17,18,41–43], whereas all workflows may be released at different time instants in dynamic scheduling [18,19,21]. The main objective of multiple DAGs-based workflows scheduling merely reduces the overall schedule length of the system or individual schedule lengths of workflows. The authors in [20] considered cost optimization of multiple DAGs-based workflows using fairness while reducing the DMR of the system. In [21–23], we studied the problem of reducing the overall schedule length while significantly reducing the DMR of the system under different situations. However, to the best of our knowledge, no related works were reported on energy management for multiple real-time workflows.

3. Models

Table 1 lists important notations and their definitions used in this study.

3.1. CPCS architecture

Fig. 1 shows the realization of CPS to CPCS [7]. We can see that the cyber part was divided into two sub cyber parts: cyber part on-device and cyber part in-cloud. That is, CPCS now operate with three key parts constituting and forming their interaction in physical and cyber world.

Fig. 2 provides the architecture of CPCS based on the description of [1,5,7,8]. CPCS contain three parts as follows:

(1) The physical part consists of sensors and actuator constellations, and interacts with the physical environments, such as lighting, temperature, water, and fertilizers.

(2) The cyber part in-cloud provides the XaaS to workflows.

(3) The cyber part on-device contains heterogeneous processors that execute the workflows.

A workflow processing in CPCS can be described as follows:

(1) A user in physical environments submits the workflow into the cyber part, and then sensors are responsible for converting the forms of workflow from continuous data to discrete data.

(2) The cloud receives converted data from sensors and provides workflow execution service. The cloud determines the workflow schedule with the forms of task mapping to processors.

(3) The heterogeneous processors execute the workflow based on the given schedule of the cloud.

(4) The heterogeneous processors finish the workflow execution and respond emergently to the cloud.

(5) The cloud sends the performing commands to actuators.

(6) The actuators convert the forms of workflow from discrete to continuous data, and actuate this data to physical environments.

Physical part and cyber part on-device are responsible for conversion and execution, and the decision on energy management and meeting real-time constraints of workflows should depend on the efficient cloud service in cyber part in cloud.

3.2. Workflow model

We let U=$\{u_1, u_2, \ldots, u_{|U|}\}$ represent a set of heterogeneous processors, where $|U|$ represents the size of set U. This study uses $|X|$ to denote the size of any set X. Multiple workflows in CPCS and is denoted as S=$\{G_1, G_2, \ldots, G_{|S|}\}$. Similar to most previous works [17,18,22,23,41–43], this study considers static scheduling and does not involve dynamic scheduling because the former is a special case of dynamic scheduling and is also a common practice. Furthermore, static scheduling is more effective than dynamic scheduling in design phase because the former can generate predictable results [22].

A workflow running on processors is represented by a DAG G_m=(N, W, E, C) [14–16]. G_m represents the mth workflows in systems.

(1) N represents a set of nodes in G_m, and each node $n_i \in N$ represents a task with different execution time values on different processors. In addition, task executions of a given workflow are assumed to be non-preemptive which is possible in many systems [16]. W is a $N \times U$ matrix, where $w_{i,k}$ denotes the execution time of n_i running on u_k with the maximum frequency [44–46]. $pred(n_i)$ 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 that has no predecessor task is denoted as n_{entry}; the task that has no successor task is denoted as n_{exit}.

(2) E is a set of communication edges, and each edge $e_{i,j}$ represents the communication message from n_i to n_j. Accordingly, $c_{i,j} \in C$ represents the communication time of $e_{i,j}$ if n_i and n_j are not assigned to the same processor. When tasks n_i and n_j are allocated to the same processor, $c_{i,j}$ becomes zero because we assume that the intra-processor communication cost can be ignored [16].

(3) In distinguishing the ambiguities among different workflows, we use $G_m.n_i$ to express the task n_i of G_m, and other attributes use the same expression. Let $LB(G_m)$ represent the lower bound of the workflow, and is the minimum schedule length of workflow G_m when all tasks are executed on the processors with the maximum frequencies by using a well-studied DAG-based scheduling algorithm (e.g., HEFT [44–46]). $D(G_m)$ represents the deadline of workflow G and should be larger than or equal to $LB(G)$. $SL(G)$ represents the generated schedule length of G_m.

Fig. 3 shows an example with three workflows of G_1, G_2, and G_3. Table 2 shows the execution time values of tasks in G_1, G_2, and G_3 of Fig. 3. The example shows ten, six, and five tasks for G_1, G_2, and G_3, respectively. This example assumes that three processors $U = \{u_1, u_2, u_3\}$ exist. Although the example is simple, it involves three processors and three workflows. Hence, this example can reflect the characteristics of multiple DAGs-based workflows on CPCS. The weight of 18 of the edge between tasks $G_1.n_1$ and $G_1.n_2$ in Fig. 3(a) represents the communication time $G_1.c_{1,2} = 18$ when $G_1.n_1$ and $G_1.n_2$ are not assigned in the same processor. The weight of 14 of $G_1.n_1$ and u_1 in Table 2(a) represents the execution time denoted by $G_1.w_{1,1} = 14$. For simplicity, all units of all parameters are ignored in the example.

Table 1
Important notations in this study.

Notation	Definition
$\|X\|$	Size of the set X
$G_m.n_i$	Task n_i of the workflow G_m
$G_m.c_{i,j}$	Communication time between the tasks n_i and n_j of the workflow G_m
$G_m.w_{i,k}$	Execution time of the task n_i on the processor u_k of the workflow G_m
$rank_u(G_m.n_i)$	Upward rank value of the task $G_m.n_i$
$LB(G_m)$	Lower bound of the workflow G_m
$D(G_m)$	Deadline of the workflow G_m
$SL(G_m)$	Final schedule length of the workflow G_m
$L(G_m)$	Laxity of the workflow G_m
$PT(G_m)$	Priority of the workflow G_m
$E(G_m.n_i, u_k, f_{k,h})$	Energy consumption of the task $G_m.n_i$ on the processor u_k with the frequency $f_{k,h}$
$E(G_m)$	Final Energy consumption of the workflow G_m
$E(S)$	Final Energy consumption of CPCS S
$DMR(S)$	Deadline miss ratio of CPCS S
$schedulable(S)$	Schedulable number of the workflows in CPCS S
$AFT(G_m.n_i)$	Actual finish time of the task $G_m.n_i$

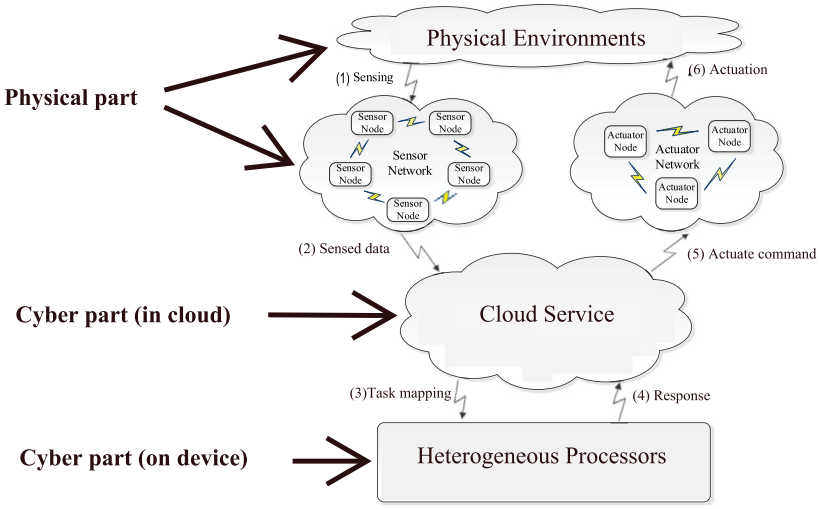


Fig. 2. Architecture of CPCS.

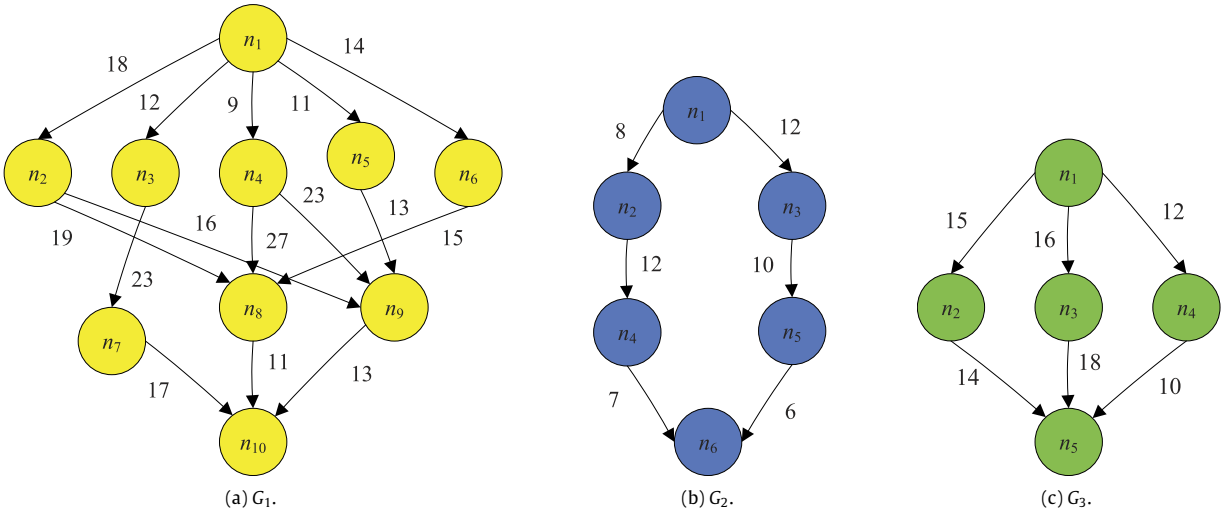


Fig. 3. Example of three workflows on CPCS.

3.3. Power and energy models

Considering the linear relationship between voltage and frequency, DVFS decreases the voltage and frequency to save energy. Similar to [44–46], we use the term frequency change to represent the process of changing the voltage and frequency simultaneously. Considering a DVFS-capable system, we also adopt the system-level power model that is widely used in [44–46], in which the

Table 2
Execution time matrixes of the workflows in Fig. 3.

(a) Execution time matrix of G_1

Tasks	$G_1.n_1$	$G_1.n_2$	$G_1.n_3$	$G_1.n_4$	$G_1.n_5$	$G_1.n_6$	$G_1.n_7$	$G_1.n_8$	$G_1.n_9$	$G_1.n_{10}$
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	6
$rank_u$	109	78	81	81	70	64	43	7360	45	15

(b) Execution time matrix of G_2

Tasks	$G_2.n_1$	$G_2.n_2$	$G_2.n_3$	$G_2.n_4$	$G_2.n_5$	$G_2.n_6$
u_1	12	9	7	13	18	15
u_2	18	15	12	15	10	10
u_3	9	11	16	18	20	8
$rank_u$	76	57	55	33	33	11

(c) Execution time matrix of G_3

Tasks	$G_3.n_1$	$G_3.n_2$	$G_3.n_3$	$G_3.n_4$	$G_3.n_5$
u_1	4	9	18	21	7
u_2	5	10	17	15	6
u_3	6	11	16	19	5
$rank_u$	62	30	41	34	6

power consumption at frequency f is given by

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

where P_s represents the static power that can only be removed by powering off the whole system. P_{ind} represents frequency-independent dynamic power that can be removed by putting the system into the sleep state. P_d represents frequency-dependent dynamic power depending 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_{ef} represents effective switching capacitance and m represents the dynamic power exponent that is not less than 2. Both C_{ef} and m are processor-dependent constants.

In this study, we assume that the system is always turned-on because turning on/off a system causes an excessive overhead. In other words, P_s is always consumed and not manageable. Similar to [44–46], this study also concentrates on dynamic power (e.g., P_{ind} and P_d) and ignore the P_s in the calculation. Less P_d does not result in less energy consumption because of the P_{ind}. That is, a minimum energy-efficient frequency f_{ee} exists [44–46] and it is denoted as

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

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

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

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

frequency-dependent dynamic power set

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

effective switching capacitance set

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

dynamic power exponent set

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

Table 3
Power parameters of processors (u_1, u_2, and u_3).

u_k	$P_{k,ind}$	$C_{k,ef}$	m_k	$f_{k,ee}$	$f_{k,max}$
u_1	0.03	0.2	2.3	0.39	1.0
u_2	0.12	0.3	2.7	0.59	1.0
u_3	0.01	1.4	2.3	0.1	1.0

minimum energy-efficient frequency set

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

and actual effective frequency set

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

We let $E(G_m.n_i, u_k, f_{k,h})$ represent the processor energy consumption of the task $G_m.n_i$ on the processor u_k with frequency $f_{k,h}$. This set is calculated by

$$E(G_m.n_i, u_k, f_{k,h}) = \left(P_{k,ind} + C_{k,ef} \times {f_{k,h}}^{m_k}\right) \times G_m.w_{i,k} \times \frac{f_{k,max}}{f_{k,h}}. \quad (2)$$

The energy consumption of the workflow G_m is calculated by

$$E(G_m) = \sum_{i=1}^{G_m.|N|} E\left(G_m.n_i, u_{pr(m,i)}, f_{pr(m,i),hz(m,i)}\right),$$

where $u_{pr(m,i)}$ and $f_{pr(m,i),hz(m,i)}$ represent the assigned processor and frequency of $G_m.n_i$, respectively. In this study, the overheads of the frequency transitions are ignored because of the negligible amount of time (e.g., 10–150 μs [16,40]).

Table 3 shows the power parameters for all the processors of the examples, where the maximum frequency $f_{k,max}$ for each processor is 1 and the frequency precision is set as 0.01. For simplicity, all the units of all parameters are ignored in the example. The minimum energy-efficient frequency $f_{k,ee}$ can be obtained for each processor using Eq. (1).

3.4. Lower bound and deadline

HEFT is a well-studied precedence-constrained task scheduling algorithm for reducing schedule length, which is based on the DAG model and with low complexity and high performance in

heterogeneous systems [14]. There are two phases to implement HEFT.

First, HEFT uses the upward rank value ($rank_u$) of a task given by Eq. (3) as the common task priority standard, where the tasks are arranged according to the decreasing order of $rank_u$. Table 2 shows the upward rank values of all tasks shown in Fig. 3 using Eq. (3):

$$rank_u(G_m.n_i) = G_m.\overline{w_i} + \max_{G_m.n_j \in succ(G_m.n_i)}\{G_m.c_{i,j} + rank_u(G_m.n_j)\}, \quad (3)$$

where $G_m.\overline{w_i}$ represents the average execution time of task $G_m.n_i$ with the maximum frequencies and is calculated by

$$G_m.\overline{w_i} = \left(\sum_{k=1}^{|U|} G_m.w_{i,k}\right)/|U|.$$

Second, the attributes $EST(G_m.n_i, u_k, f_{k,\max})$ and $EFT(G_m.n_i, u_k, f_{k,\max})$ represent the earliest start time (EST) and earliest finish time (EFT), respectively, of task $G_m.n_i$ on processor u_k with maximum frequency $f_{k,\max}$. $EFT(G_m.n_i, u_k, f_{k,\max})$ is considered the task assignment criterion in HEFT because it can satisfy the local optimal of each task. The preceding attributes are calculated by

$$\begin{cases} EST(G_m.n_i, u_k, f_{k,\max}) = \max\left\{avail[k], \max_{G_m.n_x \in pred(G_m.n_i)} \times \{AFT(G_m.n_x) + G_m.c'_{x,i}\}\right\} \\ EST(G_m.n_{\text{entry}}, u_k, f_{k,\max}) = 0 \end{cases} \quad (4)$$

and

$$EFT(G_m.n_i, u_k, f_{k,\max}) = EST(G_m.n_i, u_k, f_{k,\max}) + G_m.w_{i,k}. \quad (5)$$

$avail[k]$ is the earliest available time that processor u_k is ready for task execution, and $AFT(G_m.n_x)$ represents the actual finish time (AFT) of task $G_m.n_x$. $G_m.c'_{x,i}$ represents the actual communication time between $G_m.n_x$ and $G_m.n_i$. If $G_m.n_x$ and $G_m.n_i$ are assigned to the same processor, then $G_m.c'_{x,i} = 0$; otherwise, $G_m.c'_{x,i} = G_m.c_{x,i}$. n_i is assigned to the processor with the minimum EFT by using the insertion-based scheduling strategy. n_i is inserted into the slack with the minimum EFT. The insertion-based strategy is explained as follows: if $G_m.n_i$ can be inserted into one of the slacks of processors, then it is inserted into the slack with the minimum EFT. EFT is different from AFT because the former is the value before task assignment, whereas the latter is the value after task assignment.

Lower bound refers to the minimum schedule length of a workflow when all cores are monopolized by the workflow using a standard single DAG-based workflow scheduling algorithm. HEFT is a well-studied algorithm with low complexity and high performance that can be selected as the standard algorithm to assess the workflow. HEFT is used as the standard algorithm to explain the proposed algorithms. Other algorithms can also be easily selected and used for a simple replacement. The lower bound of the workflow G_m is the AFT of $G_m.n_{\text{exit}}$, namely,

$$SL(G_m) = AFT(G_m.n_{\text{exit}}) \quad (6)$$

where $G_m.n_{\text{exit}}$ represents the exit task of G_m. The user specifies a deadline for each workflow based on the value of the lower bound after it is obtained.

Fig. 4 shows the Gantt chart of the workflow G_1 (Fig. 3(a)) using the HEFT algorithm. The lower bound is obtained as $LB(G_1) = 80$, and the deadline is set to $D(G_1) = 100$. The arrows in Fig. 4 represent the generated communication between tasks.

Table 4 lists the properties for each workflow in Fig. 3. The task priority of each workflow can be obtained according to the descending order of $rank_u(G_m.n_i)$. Then, the lower bounds of the three workflows are as follows: $LB(G_1) = 80$, $LB(G_2) = 59$, and $LB(G_3) = 46$, respectively. The deadline of the workflow is specified under the condition $D(G_m) > LB(G_m)$. Finally, the deadlines of the three workflows are $D(G_1) = 100$, $D(G_2) = 99$, and $D(G_2) = 106$, respectively.

3.5. Problem description

Cloud service providers face the multiple workflow requests, and they need to make as many as possible DAGs completed within their deadlines [20]. We assume that CPCS service providers face the multiple workflow requests $S=\{G_1, G_2, \ldots, G_{|S|}\}$ and will be executed in heterogeneous multi-processor set $U=\{u_1, u_2, \ldots, u_{|U|}\}$. The first objective is to maximizing the number of workflows that are completed within their deadlines, namely, to minimize the DMR of CPCS:

$$DMS(S) = \frac{|S| - schedulable(S)}{|S|}, \quad (7)$$

where $schedulable(S)$ represents the number of workflows that satisfy their individual deadlines; then, the second objective is to minimize the energy consumption of the workflows that are completed within their deadlines:

$$E(S) = \sum_{m=1, SL(G_m) \le D(G_m)}^{|S|} E(G_m) = \sum_{m=1, SL(G_m) \le D(G_m)}^{|S|} \left(\sum_{i=1}^{G_m.|N|} E(G_m.n_i, u_{pr(m,i)}, f_{pr(m,i),hz(m,i)})\right), \quad (8)$$

for $\forall m : 1 \leqslant m \leqslant |S|, \forall i : 1 \leqslant i \leqslant G.|N|, u_{pr(m,i)} \in U$.

If a workflow misses its deadline, this workflow has not been able to function correctly. In [20], the workflows that exceed their individual deadlines are abandoned timely to maximize throughput. In this study, we also merely calculate the energy consumptions of workflows that satisfy individual deadlines because a workflow missing its deadline is meaningless.

4. Proposed energy management algorithms

4.1. Existing DEWTS for single real-time workflow

The state-of-the-art DEWTS algorithm [16] is proposed by merging several processors together to schedule single workflow and to minimize its energy consumption while still satisfying its real-time constraint. The core objective of DEWTS is to gradually turn off partial processors with low utilization and reassign the tasks on these turned-off processors to the turned-on processors until the schedule length exceeds the deadline. Following the processor turned-off standards of DEWTS, the processor with a small number of tasks should be turned off (if two processors have the same number of tasks, then the processor with low energy consumption is then turned off. The workflow G_1 of Fig. 3 is illustrated to explain the entire process of the DEWTS algorithm.

(1) DEWTS first invokes the HEFT algorithm for all the turned-on processors (u_1, u_2, and u_3), and the result is similar to that in Fig. 4. Schedule length $SL(G) = 80$ is obtained, and the deadline of the workflow is satisfied.

(2) Considering that the processor u_1 has the minimum task number in Fig. 4, this processor should be turned off in advance. Next, DEWTS continues to invoke the HEFT algorithm for the left turned-on processors (u_2 and u_3), and $SL(G) = 98$ is obtained (Fig. 5). In this case, the deadline of the workflow remains satisfied.

(3) Considering that the processor u_2 has the minimum task number in Fig. 5, it should be turned off. DEWTS continues to invoke the HEFT algorithm for the left processor u_3. The deadline cannot be satisfied in this case, and the final processor merging is that in Fig. 5.

(4) Considering that the merging of the processors stops at Fig. 5, DEWTS invokes the EES algorithm [15] to save energy while satisfying the deadline of the workflow on u_2 and u_3 (Fig. 6) by introducing the concept of latest finish time (LFT). The main idea

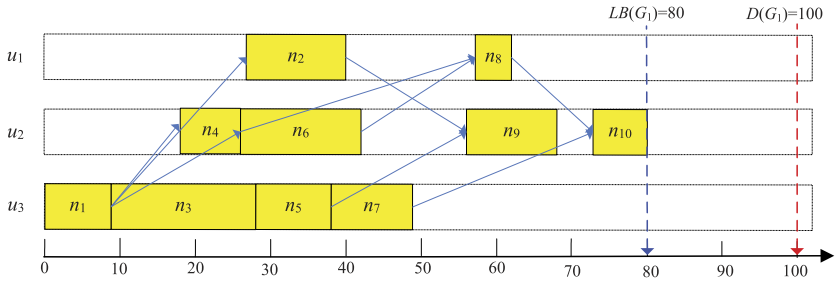


Fig. 4. Scheduling Gantt chart of the G_1 using the HEFT algorithm.

Table 4
Properties of tasks and workflows in Fig. 3.

Workflow	G_1	G_2	G_3
Task priority	$G_1.n_1, G_1.n_3, G_1.n_4, G_1.n_2, G_1.n_5, G_1.n_6, G_1.n_9, G_1.n_7, G_1.n_8, G_1.n_{10}$	$G_2.n_1, G_2.n_2, G_2.n_3, G_2.n_4, G_2.n_5, G_2.n_6$	$G_3.n_1, G_3.n_3, G_3.n_4, G_3.n_2, G_3.n_5$
$LB(G_m)$	80	59	46
$D(G_m)$	100	99	106
$L(G_m)$	20	40	60
$PT(G_m)$	1	2	3

of the EES algorithm is that the $AFT(G_m.n_i)$ obtained by the HEFT algorithm can be extended to $LFT(G_m.n_i)$ to reclaim slacks between two adjacent tasks in the same processor. The tasks are operated in descending order of $AFT(G_m.n_i)$ before using EES. $LFT(G_m.n_i)$ is calculated by

$$\begin{cases} LFT(G_m.n_i) = \min\left\{\min_{G_m.n_j \in succ(G_m.n_i)} \times \{AST(G_m.n_j) - G_m.c'_{i,j}\}, AST(G_m.n_{dn(i)})\right\} \\ LFT(G_m.n_{\text{exit}}) = D(G_m) \end{cases} \quad (9)$$

where $G_m.n_{dn(i)}$ represents the downward neighbor (DN) task of $G_m.n_i$ on the same processor. For example, we have $G_1.n_{dn(9)} = G_1.n_{10}$ and $G_1.n_{dn(5)} = G_1.n_9$ (Fig. 6). Therefore, the LFT extension of $G_m.n_i$ is restricted by its downward neighbor. For example, the AFT of $G_1.n_{10}$, as denoted with shadow, has been extended to 100 and $G_1.n_9$, denoted with shadow, has been extended to 91 while satisfying the real-time constraint (Fig. 6).

4.2. Reusable DEWTS for multiple real-time workflows

We find that DEWTS is reusable for multiple real-time workflows because the turned off processors can be turned on again to schedule the remaining workflows with the same process as Section 4.1. As mentioned in Section 3.3, considering the excessive overhead for turning on the processor, we assume processors will not be turned off but enter the sleep mode such that they can be waked up immediately when needed in this study.

If we let processors not be turned-off but be in sleep mode, then these processors can be woken up when needed. Such approach is called reusable DEWTS. However, we first should determine the workflow priority to schedule each workflow when using reusable DEWTS.

The following are the two workflow priority standards studied in existing literature: earliest deadline first (EDF) [47] and least laxity first (LLT) [48]. EDF means that the less the deadline value, the higher the priority, whereas LLT means that the less the laxity value, the higher the priority. The laxity represents the value of the deadline minus the lower bound of the workflow, and is calculated by

$$L(G_m) = D(G_m) - LB(G_m). \quad (10)$$

Table 4 shows the laxity of each workflow, namely, $L(G_1) = 20$, $L(G_2) = 40$, and $L(G_3) = 60$. According to the summary in [20], the priorities of workflows are based on LLT, namely, the less the laxity value, the higher the priority. When two workflows have the same LLT, then the priorities of workflows are based on EDF, namely, the less the deadline value, the higher the priority. The same workflow prioritizing standard is also used in this study. Each priority is denoted with a positive integer and the value of 1 represents the highest priority. Therefore, the priorities of workflows in the example are $PT(G_1) = 1$, $PT(G_2) = 2$, and $PT(G_3) = 3$, respectively.

Considering that priorities of workflows have been determined, then G_2 is scheduled by reusable DEWTS on all processors (all the processors in sleep mode are turned on). Finally, G_2 can satisfy its deadline of 99 and the AFT of $G.2.n.6$, as denoted with shadow, has been extended to 99 (Fig. 7).

Finally, G_3 is also scheduled using reusable DEWTS on all processors. However, when assigning $G_3.n_4$, its AFT is 115; that is, G_3 misses its deadline of 106 using reusable DEWTS (Fig. 8). Therefore, G_3 cannot be scheduled and should be removed.

4.3. Minimizing DMR for multiple real-time workflows

The limitation for reusable DEWTS to schedule multiple real-time workflows is that DEWTS cannot minimize DMR because high-priority workflows tend to occupy more processor execution time to save energy using DVFS, and cause low-priority workflows has less processor execution time to satisfy their deadlines. For example, G_1 and G_2 consume more processor execution time by extending the AFTs of some tasks to save energy consumption, such that G_3 cannot satisfy its deadline when assigning $G_3.n_4$ (Fig. 8). Considering that the first objective is to make as many as workflows completed within their deadlines, we should solve this objective without energy saving, such that more workflows can satisfy their deadlines. On the preceding analysis, the deadline-driven processor merging for multiple workflows (DPMMW) algorithm is presented to minimize the DMR of CPCS. The detailed algorithm of DPMMW is described in Algorithm 1.

The core idea of DPMMW is that all the workflows are assigned their individual priorities based on laxities and deadlines; DPMMW iteratively merges processors for each workflow using HEFT until the deadline of the workflow cannot be satisfied. The main steps are explained as follows:

(1) In Lines 1–4, DPMMW obtains the lower bound $LB(G_m)$ and the laxity $L(G_m)$ for each workflow.

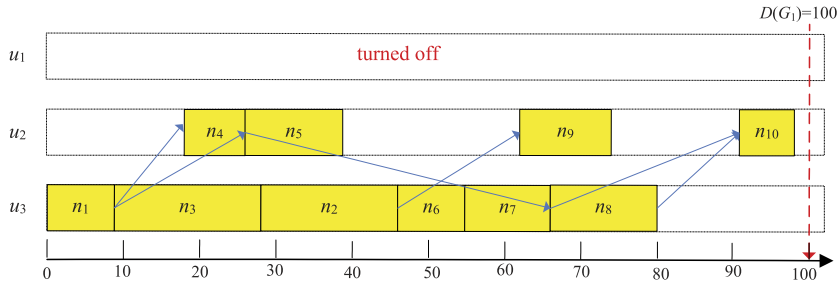


Fig. 5. HEFT-generated scheduling of G_1 (yellow colors) on u_2 and u_3 when u_1 is turned off. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

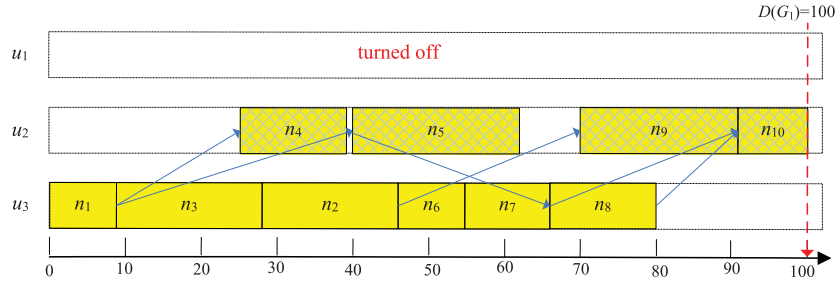


Fig. 6. EES-generated scheduling of G_1 (yellow colors) on u_2 and u_3 when u_1 is turned off. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

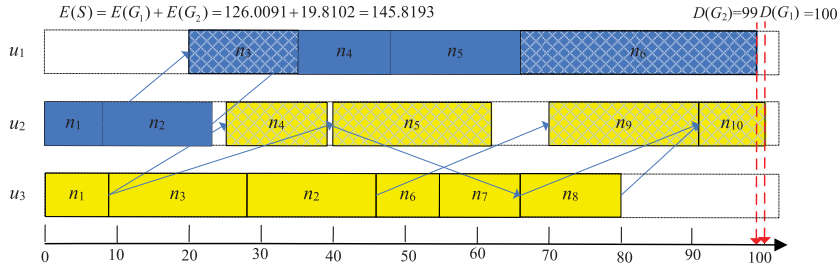


Fig. 7. Reusable DEWTS-generated scheduling of G_1 (yellow colors) and G_2 (blue colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

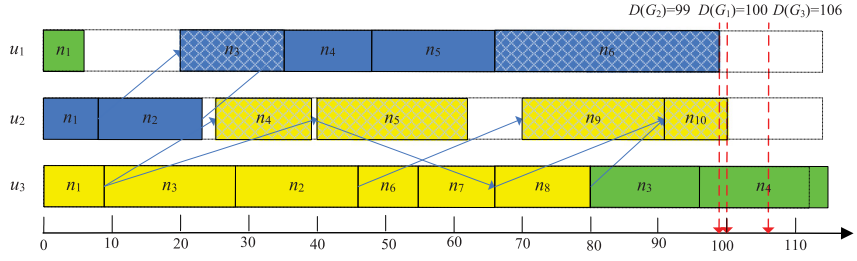


Fig. 8. Reusable DEWTS-generated scheduling of G_1 (yellow colors) and G_2 (blue colors), and G_3 (green colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(2) In Line 5, DPMMW orders workflows in a list *workflow_priority_list* in ascending order of $L(G_m)$ (if two workflows have the same laxity, then the workflows are in ascending order of $D(G_m)$).

(3) In Lines 7–26, DPMMW iteratively merges processors for each workflow using HEFT until the deadline of the workflow cannot be satisfied. In particular, all the processors should be waked up before the workflow is scheduled in Line 9. Next, DPMMW let the processor with the minimum number of tasks go to sleep mode in the iterative process based on HEFT in Line 16. When two processors have the same number of tasks, then the processors with lower energy utilization will enter sleep mode.

(4) In Line 27, DPMMW calculates the final $DMR(S)$ of CPCS.

The time complexity of the DPMMW algorithm is analyzed as follows. All the workflows can be done in $O(|S|)$ time (Line 7). Invoking the HEFT algorithm should be done in $O(|N|^2_{\max} \times |U|)$ time (Line 11), where $|N|_{\max}$ is equal to $|N|_{\max} = \max\{G_1.|N|, G_2.|N|, \ldots, G_{|S|}.|N|\}$. Traversing all processors to select one can be done in $O(|U|)$ time (Line 16). Therefore, the complexity of the DPMMW algorithm is $O(|S| \times |N|^2_{\max} \times |U|^2)$. Considering that the state-of-the-art DEWTS algorithm entails $O(|N|^2_{\max} \times |U|^2)$, reusable DEWTS is also $O(|S| \times |N|^2_{\max} \times |U|^2)$. Therefore, DPMMW has the same time complexity with reusable DEWTS.

Figs. 5 and 9–10 show the Gantt charts in scheduling the motivating example using the DPMMW algorithm.

Algorithm 1 The DPMMW Algorithm

Input: $U = \{u_1, u_2, ..., u_{|U|}\}, S = \{G_1, G_2, ..., G_{|S|}\}$
Output: $DMR(S)$ and related values
for ($m \leftarrow 1$; $m \leq |S|$; m++) **do**
  Obtain the lower bound $LB(G_m)$ using the HEFT algorithm;
  Obtain the laxity $L(G_m)$ using Eq. (10);
end for
Order workflows in a list *workflow_priority_list* according to ascending order of $L(G_m)$ (if two workflows have the same laxity, then the workflows are ordered according to ascending order of $D(G_m)$);
$schedulable(S) \leftarrow 0$;
while (*workflow_priority_list* is not null) **do**
  $G_m \leftarrow$ *workflow_priority_list.out*();
  Wake up all the processors;
  while (true) **do**
    Attempt to invoke the HEFT algorithm on all active processors to obtain the schedule length $SL(G_m)$;
    if ($SL(G_m) \leq D(G_m)$) **then**
      Mark G_m as schedulable;
      $schedulable(S)$++;
      Keep the previous HEFT-generated scheduling results;
      Let the processor (excluding the processors that has been scheduled by previous workflows) with the minimum number of tasks go to the sleep mode (if two processors have the same number of tasks, then the processors with lower energy utilization will go to the sleep mode);
      if (no processors can go to the sleep mode) **then**
        Discard the previous HEFT-generated scheduling of G_m;
        break;
      end if
    else
      Discard the previous HEFT-generated scheduling of G_m;
      break;
    end if
  end while
end while
Calculate $DMR(S)$ using Eq. (7);
return;

(1) G_1 is scheduled first because it has the highest priority. In this case, the results using DPMMW are equivalent to that using processor merging-based HEFT (Fig. 5 of Section 4.2, where $SL(G_1) = 98$, which is less than $D(G_1) = 100$).

(2) Next, G_2 has the second highest priority, and it is scheduled after G_1 is finished. The results using DPMMW are shown in Fig. 9, where $SL(G_2) = 62$, which is less than $D(G_2) = 99$.

(3) Finally, G_3 has the lowest priority, and it is scheduled after G_2 is finished. The results using DPMMW are shown in Fig. 10, where $SL(G_3) = 91$, which is less than $D(G_2) = 106$.

As can be seen, DEWTS can only satisfy the deadlines of two workflows (Fig. 8), whereas DPMMW can satisfy the deadlines of all three workflows (Fig. 10).

4.4. Minimizing energy consumption for multiple real-time workflows

After the first objective has been solved by using DPMMW (Algorithm 1), the second objective is to minimize the energy consumption of the workflows that completed within their deadlines using DVFS. This objective is then solved by reusing EES to save energy consumption. Because the original EES is merely for single workflow, we extend it and propose EES for multiple workflows (EESMW) algorithm. Similar to EES, all the tasks are operated in descending order of $LFT(G_m.n_i)$ of Eq. (9) using EESMW.

Considering that more workflows have been scheduled using DPMMW than those using reusable DEWTS in solving the first objective, the energy consumption saving using the EESMW may be less than those of using reusable DEWTS (Fig. 7) to solve the second objective. For example, as shown in Fig. 11, many tasks (denoted with shadows) have extended individual AFTs to LFTs (calculated by Eq. (9)) in the same processors by invoking the DPMMW&EESMW algorithm. However, the generated energy consumption using DPMMW&EESMW reaches 155.8397, which is larger than 145.8193 using reusable DEWTS.

Based on the results of Fig. 11, reclaiming slack times on the same processor for each task using EESMW is not energy-efficient for CPCS with multiple workflows. Using the EESMW algorithm has a major limitation. That is, only a small part of tasks near the exit task can be extended to save energy using DVFS on the same processor (e.g., $G_1.n_{10}$, $G_3.n_5$, $G_3.n_4$, $G_1.n_9$, $G_2.n_6$, $G_2.n_5$, $G_1.n_5$, and $G_3.n_2$ in Fig. 11). In other words, EESMW is a typical local approach, and more tasks could be optimized by moving them to other processors using a global approach. In the following, the global approach is presented in detail.

In [15,16], the LFT is defined (Eq. (9)) on the same processor to minimize dynamic energy consumption. However, the task may be moved to another processor and generates less energy consumption without violating the precedence constraints of all the workflows tasks. Therefore, each task should have individual LFTs on different processors. Meanwhile, the task can only be inserted to the processor slacks because such task cannot change the ASTs and AFTs of other tasks. Therefore, $slack_{k,m,i}$ represents the maximum slack on the processor u_k, in which $G_m.n_i$ can be inserted into and calculate the LFTs of each task on the slacks of different processors:

$$LFT(G_m.n_i, slack_{k,m,i}) = \min\left\{\min_{G_m.n_j \in succ(G_m.n_i)} \times \left\{AST(G_m.n_j) - G_m.c'_{i,j}\right\}, FT(slack_{k,m,i})\right\}, \quad (11)$$

where $FT(slack_{k,m,i})$ represents the finish time of the slack $slack_{k,m,i}$. In contrast to Eq. (9), the downward neighbor task of n_i on the same processor is replaced by $FT(slack_{k,m,i})$ in Eq. (11). The following two important points should be noted:

(1) All the tasks are operated according to a descending order of $AFT(G_m.n_i)$ obtained using DPMMW. Therefore, the task order for energy consumption is $G_1.n_{10}$, $G_3.n_5$, $G_3.n_4$, $G_1.n_8$, $G_1.n_9$, $G_1.n_7$, $G_3.n_3$, $G_2.n_6$, $G_1.n_6$, $G_2.n_5$, $G_1.n_2$, $G_2.n_4$, $G_1.n_5$, $G_2.n_3$, $G_1.n_3$, $G_1.n_4$, $G_2.n_2$, $G_3.n_2$, $G_1.n_1$, and $G_2.n_1$, $G_3.n_1$.

(2) Some processors may have not such slacks for several tasks. For example, the task $G_1.n_{10}$ cannot be inserted into the slacks of u_1 and u_3 (Fig. 10) because of the strict limitation of the precedence constraints with its predecessors, and the assignments of other tasks cannot be changed. Finally, $G_1.n_{10}$ extends its LFT from 98 to 100 in u_2.

The second optimized task $G_3.n_5$ can be inserted on all processors without violating precedence constraints with its predecessors. In the following, $G_3.n_5$ is taken to explain the details of the global energy consumption optimization process:

(1) The LFTs of $G_3.n_5$ on all slacks can be obtained as

$$\begin{cases} LFT(G_3.n_5, slack_{1,3,5}) = 106 \\ LFT(G_3.n_5, slack_{2,3,5}) = 106 \\ LFT(G_3.n_5, slack_{3,3,5}) = 106. \end{cases}$$

(2) To calculate the possible generated energy consumption in the $slack_{k,m,i}$, the EST of $G_m.n_i$ on the $slack_{k,m,i}$ should be obtained as follows:

$$EST(G_m.n_i, slack_{k,m,i}) = \max\left(ST(slack_{k,m,i}), \max_{n_x \in pred(G_m.n_i)} \times \left\{AFT(G_m.n_x) + (G_m.c'_{x,i})\right\}\right), \quad (12)$$

where $ST(slack_{k,m,i})$ represents the start time of the slack $slack_{k,m,i}$. The ESTs of $G_3.n_5$ on the slacks can be obtained as

$$\begin{cases} EST(G_3.n_5, slack_{1,3,5}) = 84 \\ EST(G_3.n_5, slack_{2,3,5}) = 100 \\ EST(G_3.n_5, slack_{3,3,5}) = 94. \end{cases}$$

(3) Next, the maximum execution time (MET) for n_i on the slack $slack_{k,m,i}$ is

$$MET(G_m.n_i, slack_{k,m,i}) = LFT(G_m.n_i, slack_{k,m,i}) - EST(G_m.n_i, slack_{k,m,i}). \quad (13)$$

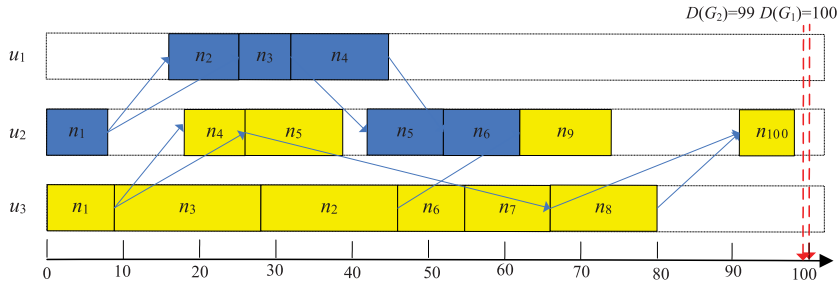


Fig. 9. DPMMW-generated scheduling of G_1 (yellow colors) and G_2 (blue colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

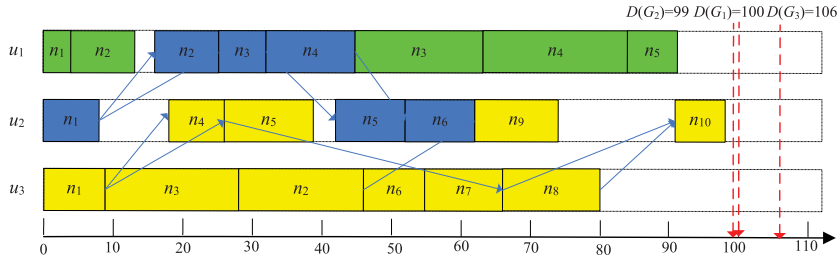


Fig. 10. DPMMW-generated scheduling of G_1 (yellow colors), and G_2 (blue colors), and G_3 (green colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

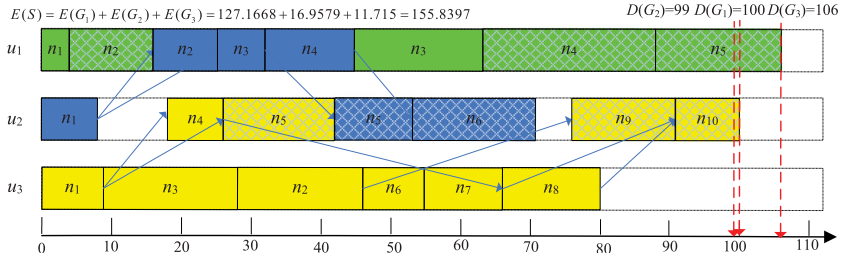


Fig. 11. DPMMW&EESMW-generated scheduling of G_1 (yellow colors), and G_2 (blue colors), and G_3 (green colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Therefore, the METs of $G_3.n_5$ on all slacks are

$$\begin{cases} MET(G_3.n_5, slack_{1,3,5}) = 22 \\ MET(G_3.n_5, slack_{2,3,5}) = 16 \\ MET(G_3.n_5, slack_{3,3,5}) = 12. \end{cases}$$

(4) Considering that each processor has the lowest frequency ($f_{k,\text{low}}$), the upper bound execution time (UBET) of n_i on the slack $slack_{k,m,i}$ is calculated by

$$UBET(G_m.n_i, slack_{k,m,i}) = \frac{f_{k,\max}}{f_{k,\text{low}}} \times G_m.w_{i,k}. \tag{14}$$

Therefore, the UBET of $G_3.n_5$ on all slacks are

$$\begin{cases} UBET(G_3.n_5, slack_{1,3,5}) = 1/0.39 \times 7 = 17.95 \\ UBET(G_3.n_5, slack_{2,3,5}) = 1/0.59 \times 6 = 10.17 \\ UBET(G_3.n_5, slack_{3,3,5}) = 1/0.1 \times 5 = 50. \end{cases}$$

(5) Next, the energy-efficient MET of $G_m.n_i$ on the slack $slack_{k,m,i}$ is calculated by

$$MET_{\text{ee}}(G_m.n_i, slack_{k,m,i}) = \min\left\{MET(G_m.n_i, slack_{k,m,i}), UBET(G_m.n_i, slack_{k,m,i})\right\}. \tag{15}$$

Therefore, the energy-efficient MET of $G_3.n_5$ on all slacks are

$$\begin{cases} MET_{\text{ee}}(G_3.n_5, slack_{1,3,5}) = \min\{22, 17.95\} = 17.95 \\ MET_{\text{ee}}(G_3.n_5, slack_{2,3,5}) = \min\{6, 10.17\} = 6 \\ MET_{\text{ee}}(G_3.n_5, slack_{3,3,5}) = \min\{12, 50\} = 12. \end{cases}$$

(6) Similarly, the minimum energy-effective frequency of n_i on the slack $slack_{k,m,i}$ is expressed as

$$f_{\text{ee}}(G_m.n_i, slack_{k,m,i}) = \frac{G_m, w_{i,k}}{MET_{\text{ee}}(G_m.n_i, slack_{k,m,i})} \times f_{k,\max}. \tag{16}$$

Therefore, we have

$$\begin{cases} f_{\text{ee}}(G_3.n_5, slack_{1,3,5}) = 7/17.95 \times 1 = 0.39 \\ f_{\text{ee}}(G_3.n_5, slack_{2,3,5}) = 6/6 \times 1 = 1 \\ f_{\text{ee}}(G_3.n_5, slack_{3,3,5}) = 5/12 \times 1 = 0.42. \end{cases}$$

(7) Next, the energy-efficient energy consumption for each slack is obtained, which is calculated by

$$E_{\text{ee}}(G_m.n_i, slack_{k,m,i}) = \left(P_{k,\text{ind}} + C_{k,\text{ef}} \times f_{\text{ee}}(G_m.n_i, slack_{k,m,i})^{m_k}\right) \times G_m, w_{i,k} \times \frac{f_{k,\max}}{f_{\text{ee}}(G_m.n_i, slack_{k,m,i})}. \tag{17}$$

For example, we have

$$\begin{cases} E_{\text{ee}}(G_3.n_5, slack_{1,3,5}) = 0.9502 \\ E_{\text{ee}}(G_3.n_5, slack_{2,3,5}) = 2.52 \\ E_{\text{ee}}(G_3.n_5, slack_{3,3,5}) = 2.4045. \end{cases}$$

(8) We select the minimum $E_{\text{ee}}(G_m.n_i, slack_{k,m,i})$ of 0.9502 on the slack $slack_{1,3,5}$ with frequency 0.39.

(9) Finally, the task $G_m.n_i$ is reassigned to the selected slack and updates the actual AFT and AST of n_i as follows:

$$AFT(G_m.n_i) = LFT(G_m.n_i, slack_{k,m,i}), \tag{18}$$

and

$$AST(G_m.n_i) = LFT(G_m.n_i, slack_{k,m,i}) - MET_{ee}(G_m.n_i, slack_{k,m,i}). \quad (19)$$

On the preceding analysis, the GESMW algorithm is presented to minimize the energy consumption of CPCS. The detailed algorithm of GESMW is described in Algorithm 2.

Algorithm 2 The GESMW Algorithm

Input: $U = \{u_1, u_2, ..., u_{|U|}\}$, $S = \{G_1, G_2, ..., G_{|S|}\}$, and values generated by DPMMW
Output: $E(S)$ and related values
1: Sort the tasks of all workfows in a list *upward_task_list* by descending order of $AFT(G_m.n_i)$ values obtained by DPMMW;
2: **while** (*upward_task_list* is not null) **do**
3: $G_m.n_i \leftarrow$ *workflow_priority_list.out*();
4: **for** (each $u_k \in U$) **do**
5: Calculate $LFT(G_m, n_i, slack_{k,m,i})$ using Eq. (11);
6: Calculate $EST(G_m, n_i, slack_{k,m,i})$ using Eq. (12);
7: Calculate $UBET(G_m, n_i, slack_{k,m,i})$ using Eq. (14);
8: Calculate $MET_{ee}(G_m, n_i, slack_{k,m,i})$ using Eq. (15);
9: Calculate $f_{ee}(G_m, n_i, slack_{k,m,i})$ using Eq. (16);
10: Calculate $E_{ee}(G_m, n_i, slack_{k,m,i})$ using Eq. (17);
11: **end for**
12: Assign $G_m.n_i$ to the $slack_{k,m,i}$with the minimum $E_{ee}(G_m, n_i, slack_{k,m,i})$;
13: $u_{pr(m,i)} \leftarrow u_k$;
14: $f_{pr(m,i),hz(m,i)} \leftarrow f_{ee}(G_m, n_i, slack_{k,m,i})$;
15: $E(G_m, n_i, u_{pr(m,i)}, f_{pr(m,i),hz(m,i)}) \leftarrow E_{ee}(G_m, n_i, slack_{k,m,i})$
16: Update $AFT(G_m.n_i)$ using Eq. (18);
17: Update $AST(G_m.n_i)$ using Eq. (19);
18: **end while**
19: Calculate $E(S)$ using Eq. (8);
20: **return**;

Compared with EESMW, the core idea of the GESMW algorithm is to reassign tasks to any processor with minimum energy consumption while satisfying deadline and precedence constraints. That is, the scope of reassignment is global but not local, which can lead to lower energy consumption than EESMW. The details of GESMW are explained as follows:

(1) In Line 1, GESMW sorts the tasks of all workflows in a list *upward_task_list* by descending order of $AFT(G_m.n_i)$ values obtained using DPMMW.

(2) In Lines 2–18, global task reassignment is implemented based on the analysis in Section 4.4.

(3) In Line 19, the new $E(S)$ is calculated.

The time complexity of the GESMW algorithm is analyzed as follows. All the tasks should be scheduled, which can be done in $O(|S| \times |N|_{max})$ time (Line 2). Traversing all processors for calculation can be done in $O(|U|)$ time (Line 4). Therefore, the complexity of the GESMW algorithm is $O(|S| \times |N|_{max} \times |U|)$. GESMW has lower time complexity than DPMMW, and DPMMW will occupy the most of the execution time when combined DPMMW&GESMW is used.

Fig. 12–14 show the Gantt charts of scheduling the motivating example using the combined DPMMW&GESMW algorithm.

(1) The tasks $G_1.n_{10}$ and $G_3.n_5$ extend individual AFTs to LFTs on fixed processors as denoted with shadows in Fig. 12. Without moving such tasks to other processors, they consume the minimum energy. When reassigning $G_1.n_8$, it is moved from u_3 to u_2 because $E_{ee}\left(G_1.n_8, slack_{2,1,8}\right) = 3.6144$, which is less than $E_{ee}\left(G_1.n_8, slack_{3,1,8}\right) = 19.74$. Furthermore, $G_1.n_8$ cannot be inserted into the slack of u_1.

(2) Considering that $G_1.n_8$ has been reassigned to u_3 from u_2, a larger slack exists in u_3. The following $G_1.n_7$ can extend its AFT to 74 on u_3 with minimum energy consumption of 7.7575 (Fig. 13). Furthermore, $G_2.n_6$ is moved from u_2 to u_3 with minimum energy consumption of 2.7963.

(3) Considering that $G_2.n_6$ has been reassigned to u_2 from u_3, a larger slack exists in u_2, the following $G_2.n_5$ can extend its AFT to 62 on u_2 with minimum energy consumption of 3.2573 (Fig. 14). Furthermore, $G_1.n_5$ also extends AFT to 45.05 on u_2 with minimum energy consumption of 4.3227. $G_3.n_2$ is also optimized in u_1.

Finally, the total energy consumption of three workflows is 131.9356, which is less than that using EESMW. A significant improvement of GESMW algorithm exists by implementing global task reassignment that can make predecessor tasks consume less energy and produce a butterfly effect. For example, moving $G_1.n_8$ is useful for $G_1.n_7$, and moving $G_2.n_6$ is useful for $G_2.n_5$ and $G_1.n_5$. In contrast, the EESMW algorithm cannot generate the above effect.

4.5. Summary of algorithms

Table 5 shows the results of the motivating example using reusable DEWTS, DPMMW&EESMW, and DPMMW&GESMW algorithms. In Table 5, the following can be observed: (1) DPMMW&EESMW and DPMMW&GESMW generate equal DMR 0, which is less than 0.33 generated by reusable DEWTS; (2) DPMMW&GESMW has the lowest energy consumption followed by reusable DEWTS and DPMMW&EESMW; and (3) all the algorithms have the time complexity of $O(|S| \times |N|^2_{max} \times |U|^2)$.

5. Experiments

5.1. Experimental metrics and workflows

The algorithms compared in the experiments are the reusable DEWTS, DPMMW&EESMW, and DPMMW&GESMW in this study. Note that reusable DEWTS is the extension of DEWTS [16] and EESMW is the extension of EES [15], whereas DPMMW&GESMW is proposed entirely in this paper. The execution parameter values are as follows [49]: 10 h $\leqslant w_{i,k} \leqslant$ 100 h, 10 h $\leqslant c_{i,j} \leqslant$ 100 h. The power parameters are as follows [45,46]: $0.1 \leqslant P_{k,s} \leqslant 0.5$, $0.03 \leqslant P_{k,ind} \leqslant 0.07$, $0.8 \leqslant C_{k,ef} \leqslant 1.2$, $2.5 \leqslant m_k \leqslant 3.0$, and $f_{k,max}$=1 GHz. All frequencies are discrete, while the precision is 0.01 GHz. All workflows are executed on simulated CPCS with 64 heterogeneous processors by creating 64 objects using Java; the source code of the experiments is provided as the supplementary material.

In order to verify the effectiveness and validity of the proposed algorithms, we use five types of workflows, namely, linear algebra [28], Gaussian elimination [14], diamond graph [28], complete binary tree [28], and fast Fourier transform [14] to compare the results of all the algorithms. Fig. 15(a)–(e) show the examples of linear algebra with the size $\rho = 5$ and the total number of tasks $|N| = \rho(\rho + 1)/2$, the Gaussian elimination with the size $\rho = 5$ and the total number of tasks $|N| = \frac{\rho^2+\rho-2}{2}$, the diamond graph with the size $\rho = 4$ and the total number of tasks $|N| = \rho^2$, the complete binary tree with the size $\rho = 5$ and the total number of tasks $|N| = 2^\rho - 1$, the fast Fourier transform with the size ρ=4 and the total number of tasks $|N| = (2 \times \rho - 1) + \rho \times \log_2 \rho$, respectively. Multiple entry tasks exist in the linear algebra workflow. We add a virtual entry task and all the actual entry tasks are set as the immediate successor tasks of the virtual entry task to adopt the workflow model of this study. Multiple exit tasks also exist in the complete binary tree and fast Fourier transform workflows. A virtual exit task should be added and all the actual exit tasks should be set as the immediate predecessor tasks of the virtual exit task.

The results for each experiment are actual values by executing one application. The reason is that experimental results with different applications in the same scale show approximate equal values.

5.2. Small-scale CPCS with multiple workflows

Experiment 1. This experiment is conducted to compare the DMR and energy consumption values of small-scale CPCS with multiple workflows for varying numbers of tasks. A small-scale CPCS means that the number of tasks for each workflow is roughly 50. In this experiment, the numbers of tasks for linear algebra, Gaussian elimination, diamond graph, complete binary tree, and

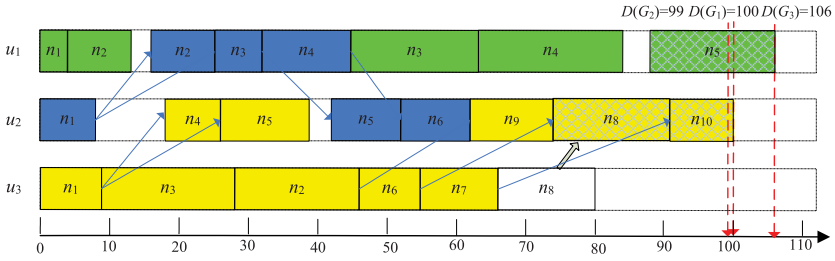


Fig. 12. DPMMW&GESMW-generated scheduling of G_1 (yellow colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

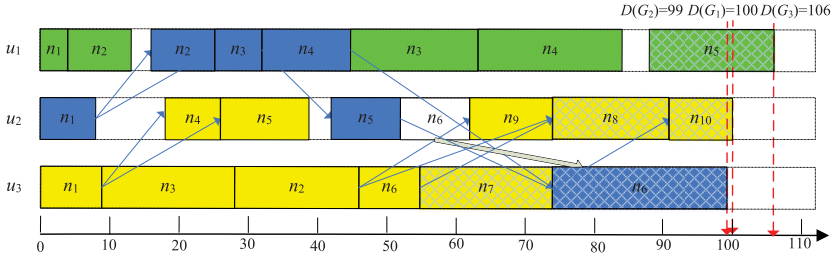


Fig. 13. DPMMW&GESMW-generated scheduling of G_1 (yellow colors) and G_2 (blue colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

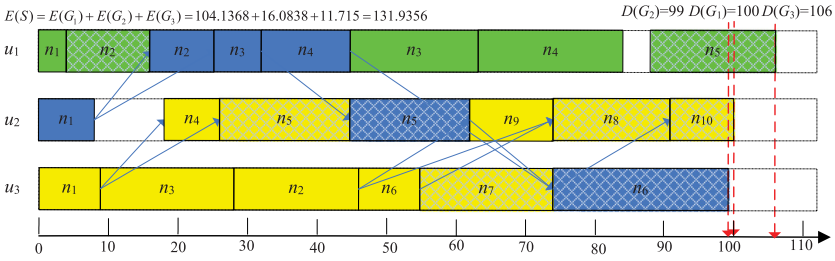
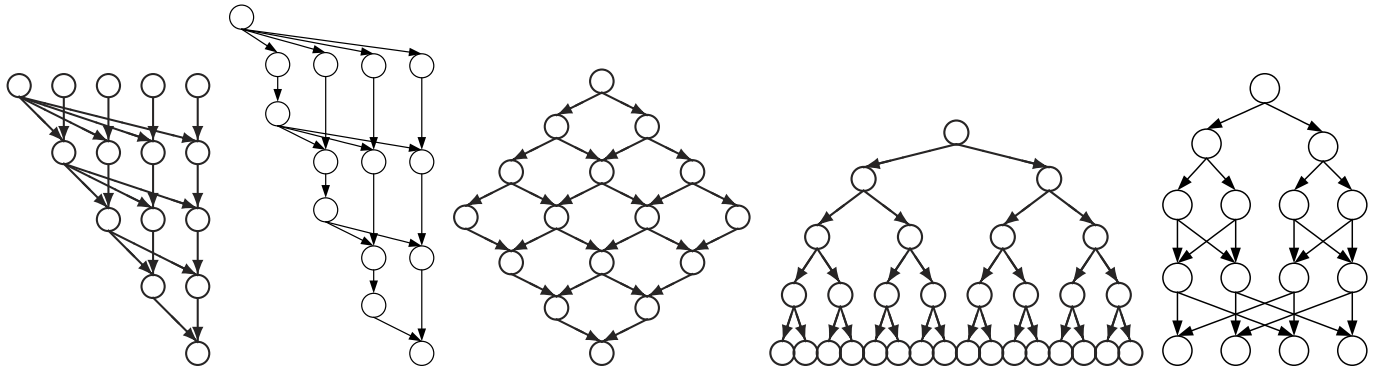


Fig. 14. DPMMW&GESMW-generated scheduling of G_1 (yellow colors) , G_2 (blue colors) and G_3 (green colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Linear algebra. (b) Gaussian elimination. (c) Diamond graph. (d) Complete binary treet. (e) Fast Fourier transform.

Fig. 15. Five different types of workflows.

Table 5
Results of the motivating example using reusable DEWTS, DPMMW&EESMW, and DPMMW&GESMW.

	Reusable DEWTS	DPMMW&EESMW	DPMMW&GESMW
$DMR(S)$	0.33	0	0
$E(S)$	145.8193	155.8397	131.9356
Time complexity	$O(\|S\| \times \|N\|^2_{max} \times \|U\|^2)$	$O(\|S\| \times \|N\|^2_{max} \times \|U\|^2)$	$O(\|S\| \times \|N\|^2_{max} \times \|U\|^2)$

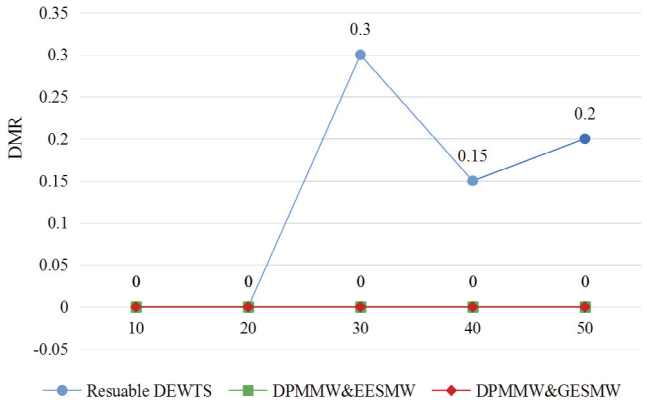


Fig. 16. DMRs of small-scale CPCS for varying numbers of tasks.

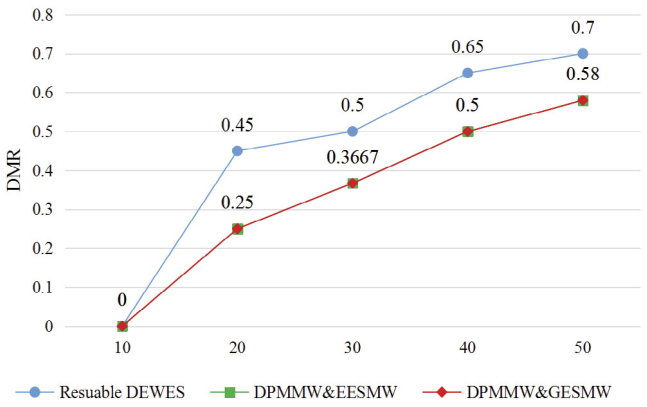
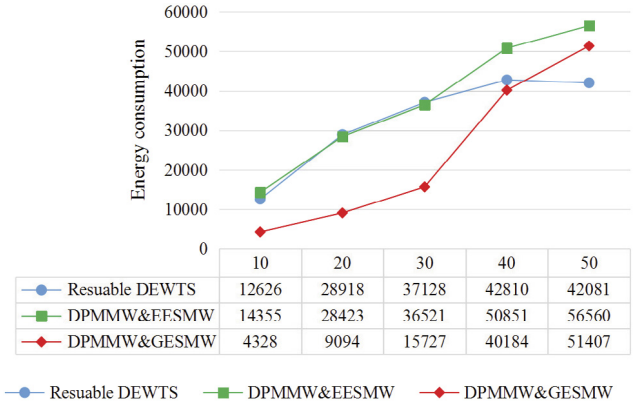
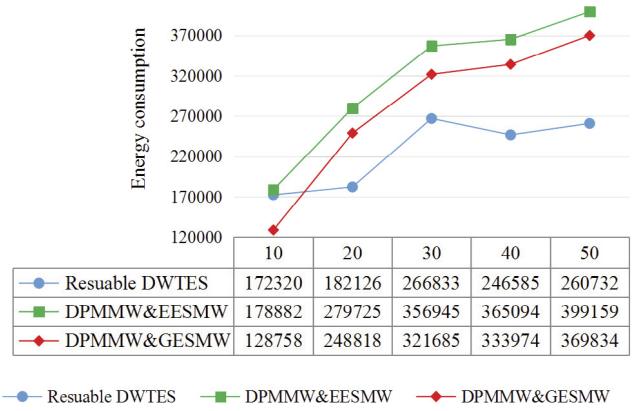


Fig. 18. DMRs of large-scale CPCS for varying numbers of tasks.



	10	20	30	40	50
Resuable DEWTS	12626	28918	37128	42810	42081
DPMMW&EESMW	14355	28423	36521	50851	56560
DPMMW&GESMW	4328	9094	15727	40184	51407

Fig. 17. Energy consumptions (unit: GWh) of small-scale CPCS for varying numbers of tasks.



	10	20	30	40	50
Resuable DWTES	172320	182126	266833	246585	260732
DPMMW&EESMW	178882	279725	356945	365094	399159
DPMMW&GESMW	128758	248818	321685	333974	369834

Fig. 19. Energy consumptions (unit: GWh) of large-scale CPCS for varying numbers of tasks.

fast Fourier transform are 55, 54, 49, 63, and 40, respectively. $D(G_m)$ is limited to the scope of $LB(G_m) = 2 \times LB(G_m)$. The number of workflows in CPCS is changed from 10 to 50 with 10 increments.

The DMR and energy consumptions are shown in Figs. 16 and 17, respectively, and the following points are summarized:

(1) DPMMW&EESMW and DPMMW&GESMW can always obtain equal DMRs of 0 in all the cases in Fig. 16. That is, all workflows can satisfy their individual deadlines in small-scale CPCS.

(2) DPMMW&EESMW and DPMMW&GESMW can always generate lower or equal DMR than (to) reusable DEWTS in all the cases in Fig. 16. When the workflow numbers reach or exceed 30, reusable DEWTS generates high DMRs of 0.15–0.3. Such results validate that reusable DEWTS occupies additional processor execution time to save energy using DVFS and cause low-priority workflows to hardly satisfy their deadlines.

(3) The energy consumptions generated by DPMMW&GESMW and reusable DEWTS are intersectant in Fig. 17. When the workflow number is less than or equal to 30, DPMMW&GESMW is superior than reusable DEWTS because they have the same DMRs 0 and the former uses the global task reassignment, which is better than the local task assignment used by the latter.

(4) When the workflow number is 40, DPMMW&GESMW generates lower DMR and less energy consumption than reusable DEWTS (Fig. 17). The result indicates that DPMMW&GESMW does not necessarily lead to less energy consumption than reusable DEWTS. The reason is still that the global approach is better than the local one.

(5) When the workflow number is 50, DPMMW&GESMW generates higher energy consumption than reusable DEWTS (Fig. 17), because more workflows satisfy individual deadlines and are scheduled using DPMMW&GESMW than using reusable DEWTS in CPCS. However, the energy consumption generated by DPMMW&GESMW remains as low as possible, and it can always generate less or equal energy consumption than DPMMW&EESMW in Fig. 17. When the workflow number is 10, DPMMW&GESMW is superior to DPMMW&EESMW by 69.85%.

(6) If DPMMW&EESMW has lower DMR than reusable DEWTS, then reusable DEWTS will have less energy consumption than DPMMW&EESMW.

5.3. Large-scale CPCS with multiple workflows

Experiment 2. Considering that CPCS are usually large-scale systems, we are interested in comparing the DMR and energy consumption values of large-scale CPCS with multiple workflows for varying numbers of tasks. Large-scale CPCS mean that the number of tasks for each workflow is roughly 1000. In this experiment, the numbers of tasks for linear algebra, Gaussian elimination, diamond graph, complete binary tree, and fast Fourier transform are 1081, 1080, 1089, 1023, 1152, respectively. We still limit $D(G_m)$ to the scope of $LB(G_m) = 2 \times LB(G_m)$. The number of workflows in CPCS is still changed from 10 to 50 with 10 increments. Figs. 18 and 19 shows the DMRs and energy consumptions of large-scale CPCS, respectively. In overall, Experiment 2 illustrates the approximate equal regular pattern as Experiment 1. The main differences are as follows:

(1) The DMR values are more larger in large-scale CPS (Fig. 18) than those in small-scale CPS (Fig. 16).

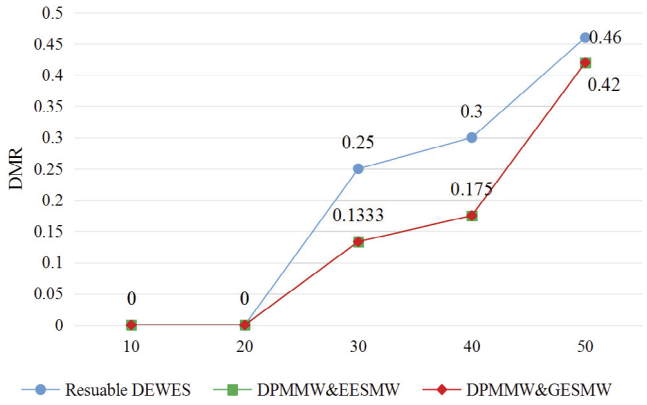
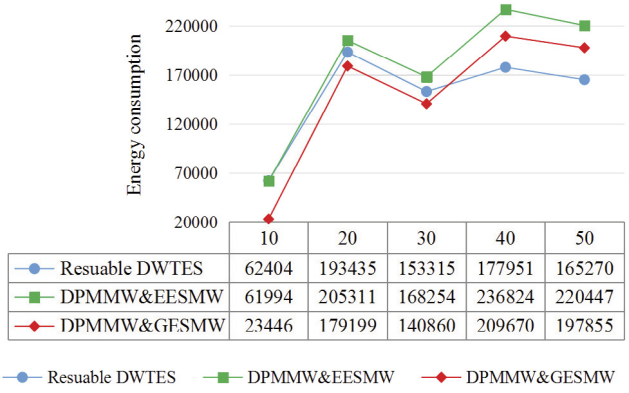


Fig. 20. DMRs of mixed-scale CPCS for varying numbers of tasks.



	10	20	30	40	50
Resuable DWTES	62404	193435	153315	177951	165270
DPMMW&EESMW	61994	205311	168254	236824	220447
DPMMW&GESMW	23446	179199	140860	209670	197855

Fig. 21. Energy consumptions (unit: GWh) of mixed-scale CPCS for varying numbers of tasks.

(2) DPMMW&GESMW always consumes less energy than reusable DEWTS when the latter generates higher DMRs than DPMMW&GESMW (Fig. 19). As a large-scale workflow with 1000 tasks will consume roughly 20 times more energy consumption than a large-scale workflow with 50 tasks, lower DMRs for DPMMW&GESMW will have more large-scale workflows to be schedule, such that larger energy consumptions are caused. Even though, global DPMMW&GESMW is always better than local DPMMW&EESMW and the energy consumptions generated by DPMMW&GESMW are still very as low as possible under lower DMRs.

5.4. Mixed-scale CPCS with multiple workflows

Experiment 3. CPCS are mixed-scale systems in which different scale workflows will be scheduled together. For such reason, the experiment is conducted to compare the DMR and energy consumption values of mixed-scale CPCS with multiple workflows for varying numbers of tasks. In mixed-scale CPCS, the numbers of tasks for linear algebra, Gaussian elimination, diamond graph, complete binary tree, and fast Fourier transform belong to the scopes of 55-1081, 54-1080, 49-1089, 63-1023, 40-1152, respectively. $D(G_m)$ is limited to the scope of $LB(G_m) = 2 \times LB(G_m)$ and the number of workflows in CPCS is still changed from 10 to 50 with 10 increments.

Figs. 20 and 21 show the DMRs and energy consumptions of mixed-scale CPCS, respectively. The results show the regular pattern as Experiment 1 and Experiment 2. DMRs and energy consumptions of mixed-scale CPCS are between those of small-scale and large-scale CPCs. DPMMW&GESMW generate low DMR and less energy consumption than reusable DEWTS when the workflow number is 30. That is, the change in mixed-scale CPCS tends to be small-scale CPCS rather than large-scale CPCS. Therefore, the proposed DPMMW&GESMW algorithm combined with the results of five different types of real workflows is effective in reducing DMR and increasing energy savings for different scale CPCS.

5.5. Summary of experiments

In summary, the reusable DEWTS, DPMMW&EESMW, and DPMMW&GESMW algorithms are compared as follows:

(1) DPMMW&EESMW and DPMMW&GESMW can always generate lower or equal DMR than (to) reusable DEWTS. That is, unlike reusable DEWTS, DPMMW does not occupy additional processor execution time to save energy using DVFS and low-priority workflows has the opportunity to satisfy their deadlines.

(2) DPMMW&GESMW can always generate less or equal energy consumption than (to) DPMMW&EESMW. Unlike EESMW, GESMW does not save energy for tasks merely on fixed processors, but it reassigns tasks to other processor slacks with minimum energy consumptions.

(3) Although reusable DEWTS is not efficient than DPMMW&GESMW in reducing DMR, the former does not necessarily lead to less energy consumption than the latter. That is, reusable DEWTS uses the local energy consumption on the fixed processor for each task, whereas DPMMW&GESMW uses the global energy consumption on all possible processors for each task.

6. Conclusions

In this study, we presented DPMMW&GESMW, an effective energy management algorithm for multiple real-time workflows on CPCS. First, DPMMW&GESMW implement lower DMRs than the state-of-the-art algorithm. Second, DPMMW&GESMW can keep as much as possible energy saving with lower DMRs. DPMMW&GESMW could also generate lower DMR and less energy consumption than the state-of-the-art algorithm in some cases. In summary, the proposed DPMMW&GESMW algorithm is effective in reducing DMR and energy saving for different scales of CPCS. Considering that CPS dynamically interacts with the physical world and dynamics is the inherent property of CPS, the energy management of dynamic multiple real-time workflows on CPCS in response to changes in the physical environments will be studied in our future work.

Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers for their constructive comments which have helped to improve the quality of the paper. This work was partially supported by the National Key Research and Development Plan of China under Grant No. 2016YFB0200405, the National Natural Science Foundation of China with Grant Nos. 61672217, 61432005, 61379115, 61402170, 61370097, 61502162 and 61502405, the CERNET Innovation Project under Grant No. NGII20161003, and the China Postdoctoral Science Foundation under Grant No. 2016M592422.

Supplemental material

The web page http://esnl.hnu.edu.cn/index.php/fgcs/ publishes the experimental codes of the paper.

References

[1] NSF, Cyber-physical systems (cps), program solicitation nsf 16-549, 2016, pp. 1–21. Website. https://www.nsf.gov/pubs/2016/nsf16549/nsf16549.htm.

[2] J. Li, Z. Ning, B. Jedari, F. Xia, I. Lee, A. Tolba, Geo-social distance-based data dissemination for socially aware networking, IEEE Access 4 (2016) 1444–1453.

[3] G. Xie, G. Zeng, Z. Li, R. Li, K. Li, Adaptive dynamic scheduling on multi-functional mixed-criticality automotive cyber-physical systems, IEEE Trans. Veh. Technol. (2017) 1–1.

[4] R. Mitchell, R. Chen, Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems, IEEE Trans. Dependable Secure Comput. 12 (1) (2015) 16–30.

[5] H. Ning, H. Liu, J. Ma, L.T. Yang, R. Huang, Cybermatics: Cyber–physical–social–thinking hyperspace based science and technology, Future Gener. Comput. Syst. 56 (2016) 504–522.

[6] J. Zeng, L.T. Yang, M. Lin, H. Ning, J. Ma, A survey: Cyber-physical-social systems and their system-level design methodology, Future Gener. Comput. Syst. (2016).

[7] S. Karnouskos, A.W. Colombo, T. Bangemann, Trends and challenges for cloud-based industrial cyber-physical systems, in: Industrial Cloud-Based Cyber-Physical Systems, Springer, 2014, pp. 231–240.

[8] N.H. Ab Rahman, W.B. Glisson, Y. Yang, K.-K.R. Choo, Forensic-by-design framework for cyber-physical cloud systems, IEEE Cloud Comput. 3 (1) (2016) 50–59.

[9] Z. Cai, X. Li, J.N. Gupta, Heuristics for provisioning services to workflows in XaaS Clouds, IEEE Trans. Serv. Comput. 9 (2) (2016) 250–263.

[10] Y. Kong, M. Zhang, D. Ye, A belief propagation-based method for task allocation in open and dynamic cloud environments, Knowl.-Based Syst. 115 (2017) 123–132. http://dx.doi.org/10.1016/j.knosys.2016.10.016.

[11] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, N. Linge, A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment, Secur. Commun. Netw. 9 (17) (2016) 4002–4012. http://dx.doi.org/10.1002/sec.1582.

[12] F. Zhangjie, S. Xingming, L. Qi, Z. Lu, S. Jiangang, Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing, IEICE Trans. Commun. 98 (1) (Jan. 2015) 190–200.

[13] Z. Fu, F. Huang, X. Sun, A. Vasilakos, C.-N. Yang, Enabling semantic search based on conceptual graphs over encrypted outsourced data, IEEE Trans. Serv. Comput. (2016) 1–1.

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

[15] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, X. Huang, Enhanced energy-efficient scheduling for parallel applications in cloud, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ccgrid 2012, IEEE Computer Society, 2012, pp. 781–786.

[16] Z. Tang, L. Qi, Z. Cheng, K. Li, S.U. Khan, K. Li, An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment, J. Grid Comput. 14 (1) (2016) 55–74.

[17] Z. Yu, W. Shi, A planner-guided scheduling strategy for multiple workflow applications, in: 2008 International Conference on Parallel Processing-Workshops, IEEE, 2008, pp. 1–8.

[18] C.-C. Hsu, K.-C. Huang, F.-J. Wang, Online scheduling of workflow applications in grid environments, Future Gener. Comput. Syst. 27 (6) (2011) 860–870.

[19] H. Arabnejad, J. Barbosa, Fairness resource sharing for dynamic workflow scheduling on heterogeneous systems, in: 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications, IEEE, 2012, pp. 633–639.

[20] W. Wang, Q. Wu, Y. Tan, F. Wu, Maximize throughput scheduling and cost-fairness optimization for multiple DAGs with deadline constraint, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2015, pp. 621–634.

[21] G. Xie, L. Liu, L. Yang, R. Li, Scheduling trade-off of dynamic multiple parallel workflows on heterogeneous distributed computing systems, Concurrency Comput.-Pract. Exp. 29 (8) (2017) 1–18. http://dx.doi.org/10.1002/cpe.3782.

[22] 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. 70 (2016) 3–14.

[23] G. Xie, G. Zeng, L. Liu, R. Li, K. Li, Mixed real-time scheduling of multiple DAGs-based applications on heterogeneous multi-core processors, Microprocess. Microsyst. 47 (2016) 93–103.

[24] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, J. Wu, Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment, J. Syst. Softw. 99 (2) (2015) 20C35.

[25] X. Zhu, C. He, K. Li, X. Qin, Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters, J. Parallel Distrib. Comput. 72 (6) (2012) 751–763.

[26] G. Zeng, Y. Matsubara, H. Tomiyama, H. Takada, Energy-aware task migration for multiprocessor real-time systems, Future Gener. Comput. Syst. 56 (2016) 220–228.

[27] K. Li, Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1484–1497.

[28] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, IEEE Trans. Comput. 61 (12) (2012) 1668–1681.

[29] K. Li, Power and performance management for parallel computations in clouds and data centers, J. Comput. System Sci. 82 (2) (2016) 174–190.

[30] K. Li, Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels, J. Parallel Distrib. Comput. 95 (2016) 15–28.

[31] Q. Tang, S.K.S. Gupta, G. Varsamopoulos, Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1458–1472.

[32] L. Parolini, N. Tolia, B. Sinopoli, B.H. Krogh, A cyber-physical systems approach to energy management in data centers, in: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems, ACM, 2010, pp. 168–177.

[33] L. Parolini, B. Sinopoli, B.H. Krogh, Z. Wang, A cyber–physical systems approach to data center modeling and control for energy efficiency, Proc. IEEE 100 (1) (2012) 254–268.

[34] S.K. Gupta, T. Mukherjee, G. Varsamopoulos, A. Banerjee, Research directions in energy-sustainable cyber–physical systems, Sustainable Comput.: Inform. Syst. 1 (1) (2011) 57–74.

[35] M. Lin, Y. Pan, L.T. Yang, M. Guo, N. Zheng, Scheduling co-design for reliability and energy in cyber-physical systems, IEEE Trans. Emerg. Top. Comput. 1 (2) (2013) 353–365.

[36] A.Y. Saber, G.K. Venayagamoorthy, Efficient utilization of renewable energy sources by gridable vehicles in cyber-physical energy systems, IEEE Syst. J. 4 (3) (2010) 285–294.

[37] P. Palensky, E. Widl, A. Elsheikh, Simulating cyber-physical energy systems: challenges, tools and methods, IEEE Trans. Syst. Man Cybern.: Syst. 44 (3) (2014) 318–326.

[38] J. Kleissl, Y. Agarwal, Cyber-physical energy systems: focus on smart buildings, in: Proceedings of the 47th Design Automation Conference, ACM, 2010, pp. 749–754.

[39] 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.

[40] 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.

[41] U. Hönig, W. Schiffmann, A meta-algorithm for scheduling multiple DAGs in homogeneous system environments, in: Proceedings of the Eighteenth IASTED International Conference on Parallel and Distributed Computing and Systems, PDCS06, 2006.

[42] H. Zhao, R. Sakellariou, Scheduling multiple DAGs onto heterogeneous systems, in: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, IEEE, 2006, p. 14.

[43] L.F. Bittencourt, E.R. Madeira, Towards the scheduling of multiple workflows on computational grids, J. Grid. Comput. 8 (3) (2010) 419–441.

[44] D. Zhu, H. Aydin, Reliability-aware energy management for periodic real-time tasks, IEEE Trans. Comput. 58 (10) (2009) 1382–1397.

[45] B. Zhao, H. Aydin, D. Zhu, On maximizing reliability of real-time embedded applications under hard energy constraint, IEEE Trans. Ind. Inf. 6 (3) (2010) 316–328.

[46] 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.

[47] T.P. Baker, An analysis of EDF schedulability on a multiprocessor, IEEE Trans. Parallel Distrib. Syst. 16 (8) (2005) 760–768.

[48] G.L. Stavrinides, H.D. Karatza, Scheduling real-time DAGs in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes, Future Gener. Comput. Syst. 28 (7) (2012) 977–988.

[49] M.W. Convolbo, J. Chou, Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources, J. Supercomput. 72 (3) (2016) 985–1012.

Guoqi Xie received his Ph.D. degree in computer science and engineering from Hunan University, China, in 2014. He was a Postdoctoral Researcher at Nagoya University, Japan, from 2014 to 2015. Since 2015 he is working as a Postdoctoral Researcher at Hunan University, China. He has received the best paper award from ISPA 2016. His major interests include embedded and real-time systems, parallel and distributed systems, software engineering and methodology. He is a member of IEEE, ACM, and CCF.

Gang Zeng is an Associate Professor at the Graduate School of Engineering, Nagoya University. He received his Ph.D. degree in Information Science from Chiba University in 2006. From 2006 to 2010, he was a Researcher, and then Assistant Professor at the Center for Embedded Computing Systems (NCES), the Graduate School of Information Science, Nagoya University. His research interests mainly include power-aware computing and real-time embedded system design. He is a member of IEEE and IPSJ.

Junqiang Jiang is currently working toward the Ph.D. degree at Hunan University, China. His research interests include scheduling in parallel and distributed systems scheduling, real-time scheduling and energy-aware scheduling.

Chunnian Fan received her Ph.D. degree in computer science from Nanjing University in 2011. Her research interests include parallel and distributed system, pattern recognition and image processing.

Renfa Li is a Professor of computer science and electronic engineering, and the Dean of College of Computer Science and Electronic Engineering, Hunan University, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. His major interests include computer architectures, embedded computing systems, cyber–physical systems, and Internet of things. He is a member of the council of CCF, a senior member of IEEE, and a senior member of ACM.

Keqin Li is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber–physical systems. He has published over 475 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Services Computing, Journal of Parallel and Distributed Computing. He is an IEEE Fellow.