Scheduling Parallel Workflow Applications with Energy-Aware on a Cloud Platform

Thanawut Thanavanich[1] and **Putchong Uthayopas** [2], Non-members

ABSTRACT

An inefficient energy consumption of computing resources in a large cloud datacenter is a very important issue since the energy cost is now a major part of the operating expense. In this paper, the challenge of scheduling a parallel application on a cloud platform to achieve both time and energy efficiency is addressed by two new proposed algorithms Enhancing Heterogonous Earliest Finish Time (EHEFT) and Enhancing Critical Path on a Processor (ECPOP). The objective of these two algorithms is to reduce the energy consumption while achieving the best execution makespan. The algorithms use a metric that identifies and turns off the inefficient processors to reduce energy consumption. Then, the application tasks are rescheduled on fewer processors to obtain better energy efficiency. The experimental results from the simulation using real-world application workload show that the proposed algorithms not only reduce the energy consumption, but also maintain an acceptable scheduling quality. Thus, these algorithms can be employed to substantially reduce the operating cost in a large cloud computing system.

Keywords: Task Scheduling, Energy-aware, Cloud Computing, Energy Efficiency

1. INTRODUCTION

With rapid growth worldwide in information technology demands during the last decades, there is an emerging trend in the use of public utility computing services based on the cloud computing platforms. Cloud computing is an emerging technology that enables a massive number of computer servers to act as a single system. This technology helps deliver a massive and scalable computing infrastructure for modern computing needs. In general, a cloud datacenter is composed of a large pool of computing resources which can be utilized to execute a cloud application with a reasonable time. One of the main issues in operating a cloud data center is the energy cost reduction. Recent study [1] shows that the power consumption in large-scale datacenter is now account for as much as 0.5 % of the world's total power usage. For example, a datacenter housing 1,000 racks can consume approximately 10 MW of electricity [2]. Junaid Shuja et al. [3] conducted a study and found that the cost of power consumption in a datacenter has doubled every five years. By reducing the energy used to execute the application may lead to performance degradation (e.g., longer completion time), since the speed of processors also depends on energy and clock frequency. Thus, the energy-efficiency issue and task execution performance must be addressed together by the scheduler. Normally, there are two approaches to reduce energy consumption in a large-scale datacenter [4] [5] . First, a static approach can be used by upgrading the hardware components in the datacenter. Second, the dynamic approach that takes into consideration the energy consumption and computing power must be used. Many efforts in task scheduling focused on a single performance goal such as minimizing the application execution time (i.e., the schedule length or makespan). By deploying an energy-aware task scheduling, the cloud platform can be utilized more efficiently while the operating cost is substantially reduced.

In this paper, we propose two enhanced energy-aware scheduling algorithms, called *Enhancing Heterogonous Earliest Finish Time (EHEFT)* and *Enhancing Critical Path on a Processor (ECPOP)*. Both algorithms take into account the makespan conservation and energy reduction. A metric called *ratio of effectiveness* (**RE**) is proposed that measure a level of virtue for executing a task on a processor, and how efficient the energy is consumed. The main difference between the proposed scheduling algorithms and previous works are the measurement of the utilization of task assigned to each processor.

This paper is organized as follows. First, the reviews of the related works are presented.Then, the cloud system model, cloud application, energy consumption, and scheduling model are defined followed by the presentation of the proposed algorithms. The experiment has been conducted using simulation and the results are given with discussion in the next section. Finally, the summary of the work has been provided.

2. RELATED WORK

The energy consumption reduction in various systems such as cluster, grid, and cloud is an important problem that was addressed in many literatures [4]

Manuscript received on October 30, 2014.
Final manuscript received December 5, 2014.

[1,2] The author is with Computer Engineering Department, Faculty of Engineering, Kasetsart University, Bangkok, Thailand., E-mail: thanawut.cru@gmail.com and pu@ku.ac.th

[6] [7] [8]. The first approach, the Static Power Management (SPM), requires low-power hardware equipment e.g., mobile or handheld device. The second approach, the Dynamic power management (DPM) techniques, is based on the improvement of current resource utilization and the management of application workloads. The DPM relies on software techniques, power-scalable processor, and memory components. In recent literature, many effective power-scalable techniques exist for reducing the energy consumption of a processor [4] [10]. One of the most used techniques to reduce the power consumption of a processor is called Dynamic Voltage/Frequency Scaling (DVFS) [10] [11] [12] [13] [14] [15] . This technique enables a processor to operate at different voltage level. By lowering the level of voltage during the idle time, a significant reduction in power consumption can be achieved. For software component, a suitable scheduling algorithm can increase the resource utilization while adjusting a proper operating power for processors was proposed in [11] [15] [16] [17] [18] [20] [21].

Scheduling of parallel applications presented as a set of precedence-constrained tasks has been studied with different environments e.g., single processor system [20] [22], homogeneous system [11] [15] and heterogeneous system [11] [16] [18] [19]. In general, task scheduling algorithm for those environments can be classified into two groups: static and dynamic scheduling [23]. In static scheduling, task information (i.e. computation cost of tasks and communication cost between tasks) must be known prior to the task execution using some estimation technique. In contrast, task information in dynamic scheduling can be obtained when the task is executed. However, both scheduling approaches focused on the minimization of the application execution time (called schedule length or makespan) [11] [16] [17] [18] [19] [23] [24] [25] [26] [27] [28] [29]. In recent works [11] [15] [16] [17] [18] [19] [20] [22] [29], the reduction of energy usage was taken into consideration as another important objective of the task scheduling. For example, a concept of makespan conservation with energy reduction technique was proposed [17]. This technique is based on adopting the slack reclamation to achieve the energy reduction. The concept of slack management was proposed to enhance a utilization of task execution within idle phase in processors enabling DVFS [13]. The slack time occur in processor that executes a task with earlier completion. Nevertheless, the application of scheduling with DVFS may results in a lower power consumption but longer execution time.

Mainly, two approaches, namely heuristic [17] [30] and metaheuristic[19] [29] [31] [32] [33] were applied to address this problem. Most works applied heuristic methods for scheduling a set of tasks. The heuristic method is one of the most commonly used approaches to schedule a parallel application. This method can be classified into three main techniques [25], list scheduling [34], clustering [35], and task duplication [5]. Another method is metaheuristic approach that uses random choice to guide an approximate solution. There are many well-known algorithms such as Genetic Algorithm (GA) [19] [29], Artificial Bee Colony (ABC) [32], Ant Colony Optimization (ACO) [33], and Particle Swarm Optimization (PSO)[31]. Although meta-heuristic approach can generates good solution, they usually spend much longer time to get a solution as a system become very large.

Energy-aware scheduling algorithms proposed in many previous works [11] [13] [15] [16] [17] [18] [19] [20] [21] [28] [29] [34] [36] took into account the performance loss due to the decreasing computing power. To effectively balance both objectives a concept of makespan conservation with energy reduction technique is proposed in [17] [18]. The HEFT and CPOP algorithms were adapted for scheduling with energy-aware in many works [16] [17] [18] [37]. We found that both scheduling algorithms with slack management technique still consume energy while waiting for data from the precedence task. Those scheduling algorithms aim to schedule tasks within a deadline and also reduce the power consumption. Our proposed scheduling algorithms focus on maintaining the makespan of task scheduled and reducing energy consumption at the same time. Moreover, the propose methods can be applied to various environments and system with different power management technique.

3. MODELS

3.1 Cloud System Model

A cloud data center considered in this work consists of a large pool of heterogeneous machines viewed as services. The data center consists of a set P of fully connecting p processors. Each processor $p_j \in P$ is DVFS-enabled. Hence, the processors can operate with different voltage levels and clock frequencies. We define the supply voltage on processor p_j as set V_j of v and the clock frequency on processor $p_j$as set F_j of f. While the supply voltage operates at level v_l, the clock frequency also performs in f_l. A processor in an idle state of execution and waiting data of precedence task will operate at a lowest voltage level (v_{low}) . In this paper, we assume that the overhead of the frequency transition is negligible. In addition, the inter-processor communication is assumed to be at the same speed for all processors. It is also assumed that data can be exchanged among processors while a task is executing. In this work, one physical machine will host only one virtual machine. This virtual machine is implemented using virtualization tool, i.e. Linux KVM, VMware, Xen, Parallel Desktop, and Virtual Box.

3.2 Cloud Application Model

A cloud application is a workflow represented by a directed acyclic graph *(DAG)*, for example, the graph shown in Fig. 1 . Task graph $G = (T, E)$ consists of a set of vertices T and edges E. Each vertex represents a task tagged with the computation cost of the application. Each edge represents the precedence constraints of the task. The weight of each edge is the communication cost between two tasks. In general, a task without precedence is called an **entry task** (t_{entry}) and a task without successor is called an **exit task** (t_{exit}). The **schedule length** is defined as the finish time of the latest task. The **critical path** *(CP)* path is defined as the longest execution path of the scheduled task graph.

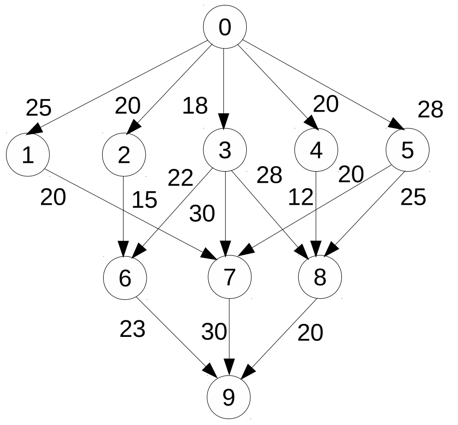


Fig.1: *A task graph*

Table 1: *Computation cost on each processor*

$Task_i$	P_0	P_1	P_2	P_3	P_4	$\bar{w_i}$
0	20	16	15	13	12	15.2
1	15	19	18	17	10	15.8
2	11	13	19	15	20	15.6
3	13	8	17	16	21	15
4	12	13	10	15	14	12.8
5	13	16	20	9	18	15.2
6	7	15	11	8	12	10.6
7	5	11	14	13	7	10
8	18	12	20	15	21	17.2
9	21	10	16	15	10	14.4

The estimated task execution of a given application denoted as W represents the computation cost matrix $t_i \times p_j$. Each w_{ij} is an execution time of task t_i on processor p_j (for example, the execution times in Table 1). An average computation cost of task t_i can be defined as $\bar{w_i}$. The task communication denoted as c_{ij} represents the communication cost between task t_i and t_j . The communication cost is ignored when both tasks are allocated to the same processors.

3.3 Energy Computing Model

The energy consumption of a task execution on any processor E_{total} composed of the dynamic energy consumption $E_{dynamic}$, static energy consumption E_{static}, and energy spent execute empty task E_{idle}. In this work, the static energy consumption is ignored since the dynamic power dissipation is the most significant factor of the energy consumption [16] [17]. Therefore, the total energy consumption E_{total} is given by $E_{total} = E_{dynamic} + E_{idle}$. The dynamic power dissipation $P_{dynamic}$ is computed as $P_{dynamic} = ACV_{jl}^2 f_{jl}$ where A is the number of switches per clock cycle,C is the total capacitance load, V_{jl} is the supply voltage at level l on processor p_j , and f is the operating frequency that operated the supply voltage at level l on processor p . The parameters A and C , which are device related constants, depend on each device capacity. Thus, the dynamic energy consumption is computed as $E_{dynamic} = P_{dynamic} \Delta w_{ij}$ where w_{ij} is the computation cost of task t_i on the processor p_j (the amount of time used to execute task). In the task execution period, we assume that the processor p_j operates at the highest level of supply voltage $v_{j,high}^2$ and the highest frequency $f_{j,high}$. On the other hand, the processor scales down the voltage and frequency to lowest supply voltage $v_{j,low}^2$, and lowest frequency $f_{j,low}$, for idle period. The energy consumption for all the idle period E_{idle} can be represented as $E_{idle} = ACV_{j,low}^2 f_{j,low} \Delta w_{idle,j}$ where $w_{idle,j}$ is an idle time slot to execute the empty task on processor p_j.

4. ENERGY-AWARE SCHEDULING

In this work, the task scheduling problem is defined as the allocation of a set T of t tasks to a set P of p processors while minimizing the schedule length and energy consumption. After the scheduling of t tasks is completed, the *schedule length* or *makespan* is obtained through the completion time t_{exit}. The question addressed is how to obtain a scheduling which lower the energy usage while preserving the makespan as much as possible. Fig. 2(a) and 2(b) show the task scheduling results given the task graph in Fig. 1 scheduled by HEFT and CPOP algorithms, respectively. Since the primary performance goal of the HEFT and CPOP are to assign all tasks to minimize the completion time, the scheduling will not consider the reduction of processors to increase the energy efficiency. By taking some shutting some processors, there is the possibility of saving energy while slightly increasing the makespan (as shown in Fig. 2(c) and 2(d)). The result shows that the schedule has length equal to 114 for EHEFT and 118 for ECPOP. This work tries to identify an opportunity of shutdown some processors for energy saving. Although, there are a few works that use different approaches for saving the energy (e.g. [16] [17]), this work uses the schedule length and energy consumption of HEFT and CPOP as based line algorithm.

4.1 Proposed Scheduling Algorithms

In this work, the scheduling algorithm called EHEFT and ECPOP is proposed (as shown in Table. 2 and 3). The main idea of the algorithm is to reduce the energy consumption by shutting down a set of processors that inefficiently execute the tasks. All of the processors allocated to a set of tasks are evaluated by applying the ratio between active time and makespan as criteria. EHEFT will try to aggressively reduce the power consumption by allowing makespan to be longer. ECPOP algorithm will use critical path to maintain the make span. Thus, the performance is better but the energy usage of the application is higher than EHEFT. The following sections will describe in detail the concept of both proposed algorithms.

Table 2: *EHEFT*

Algorithm1 :	*EHEFT(Enhancing HEFT)*
Input :	$G = (T, E)$ and a set of p processor
Output :	EHEFT scheduled set of G onto set of P
Phase1 :	Task scheduling with HEFT onto set P
Phase2 :	Compute RE of the assigned processor with Algorithm 3: finding a set of inefficient processor Shutdown a set of inefficient processor
Phase3 :	Rescheduling task with HEFT

4.2 EHEFT Scheduling Algorithm

The first algorithm, **EHEFT** is an enhancement of HEFT algorithm and consists of three phases as follows.

4.2.1 Phase 1: Task Scheduling

Heterogeneous Earliest-Finish Time (HEFT) is used to allocate tasks to processors. The result of the HEFT is the schedule of a given task graph that minimize the makespan for this graph.

4.2.2 Phase 2: Discovering Inefficient Processor

This phase discovers the inefficient processors using a performance metric called *ratio of effectiveness (RE)* . The RE value can be used to measure the utilization of task execution on a processor. Let RE_j denotes the RE value on processor p_j. The RE value is obtained by calculating the ratio of total active time of a processor and schedule length as given in 1.

$$RE_j = \frac{\Sigma_{i \in p_j} w_{ij}}{Makespan} \qquad (1)$$

Let w_{ij} denote an active time slot which runs a task t_i on the processor p_j. The output of this step is the sorted RE list in an ascending order. For a processor with a low RE value, most of the energy is consumed by idle time slot not the task execution. Therefore, these low RE processor can be shutdown to save energy. The process mentioned is summarized in Table 4.

4.2.3 Phase 3: Rescheduling

The EHEFT uses the set of the assigned processors that exclude a set of inefficient processors. Let P' be the set of processors after a set of inefficient processors P_{ineff} were shutdown. After the new set of processor has been identified, the same task graph is rescheduled again using HEFT using P' . For example, EHEFT is applied to schedule tasks in Fig. 2(c). The set of processors is used that is $\{p_0, p_1, p_2, p_3, p_4\}$.

Table 3: *ECPOP*

Algorithm2 :	*ECPOP(Enhancing CPOP)*
Input :	$G = (T, E)$ and a set of p processor
Output :	ECPOP scheduled set of G onto set of P
Phase1 :	Task scheduling with CPOP onto set P
Phase2 :	Compute RE of the **non-critical processor** with Algorithm 3: finding a set of inefficient processor Shutdown a set of inefficient processor
Phase3 :	Rescheduling task with CPOP

4.3 ECPOP Scheduling Algorithm

In this section, another energy-aware scheduling called **ECPOP** is proposed. The goal is to reduce the energy consumption by increasing the utilization of task execution while conserving the makespan. In this algorithm, *critical processor* is defined as a processor that executes a task that is on the critical path. On the other hand, any processor that is assigned with non-critical task is called *non-critical processor* . A set of non-critical processors are evaluated for the utilization. The metric used is *RE (ratio of effectiveness)* which is the same as in EHEFT algorithm. The makespan conservative technique applied to the ECPOP is to shutdown the non-critical processors only. The ECPOP algorithm consists of three phases as follows.

4.3.1 Phase 1: Task Scheduling

First, the traditional Critical Path on a Processor (CPOP) algorithm is used to allocate the critical tasks to the processors which have the minimum total execution time. The result is a schedule of the critical tasks on a set of processors.

4.3.2 Phase 2: Discovering Inefficient Processor

Let P_{nCP} denote a set of processors that execute non-critical task. Hence, P_{nCP} is given by $P_{nCP} = P - P_{CP}$ where P_{CP} is a set of processor that executes the critical tasks. After determining the set of P_{nCP}, the RE value of each processor is computed

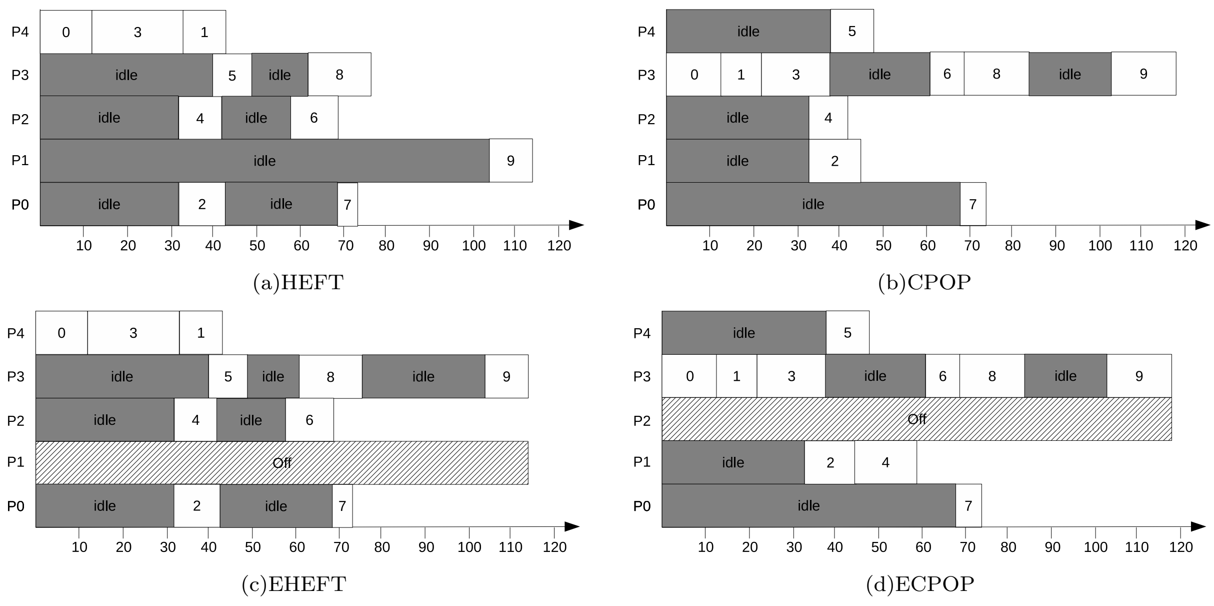


Fig.2: *Scheduling of task in Figure. 1*

to identify the inefficient processor that can be shutdown without increasing the makespan. Thus, the RE measure is given in 2.

$$RE_j = \frac{\Sigma_{i \in p_j^{nCP}} w_{ij}}{Makespan} \tag{2}$$

Let w_{ij} denote an active time slot which runs a task t_i on the processor p_j^{nCP}. The output of this step is the sorted RE list in an ascending order. Next, a set of tasks allocated on the lowest RE processor are assigned to the processor p_k^{nCP} where RE^{lowest} . The reassigned step is repeated till a set of tasks cannot be allocated.

Table 4: *Finding a set of inefficient processor*

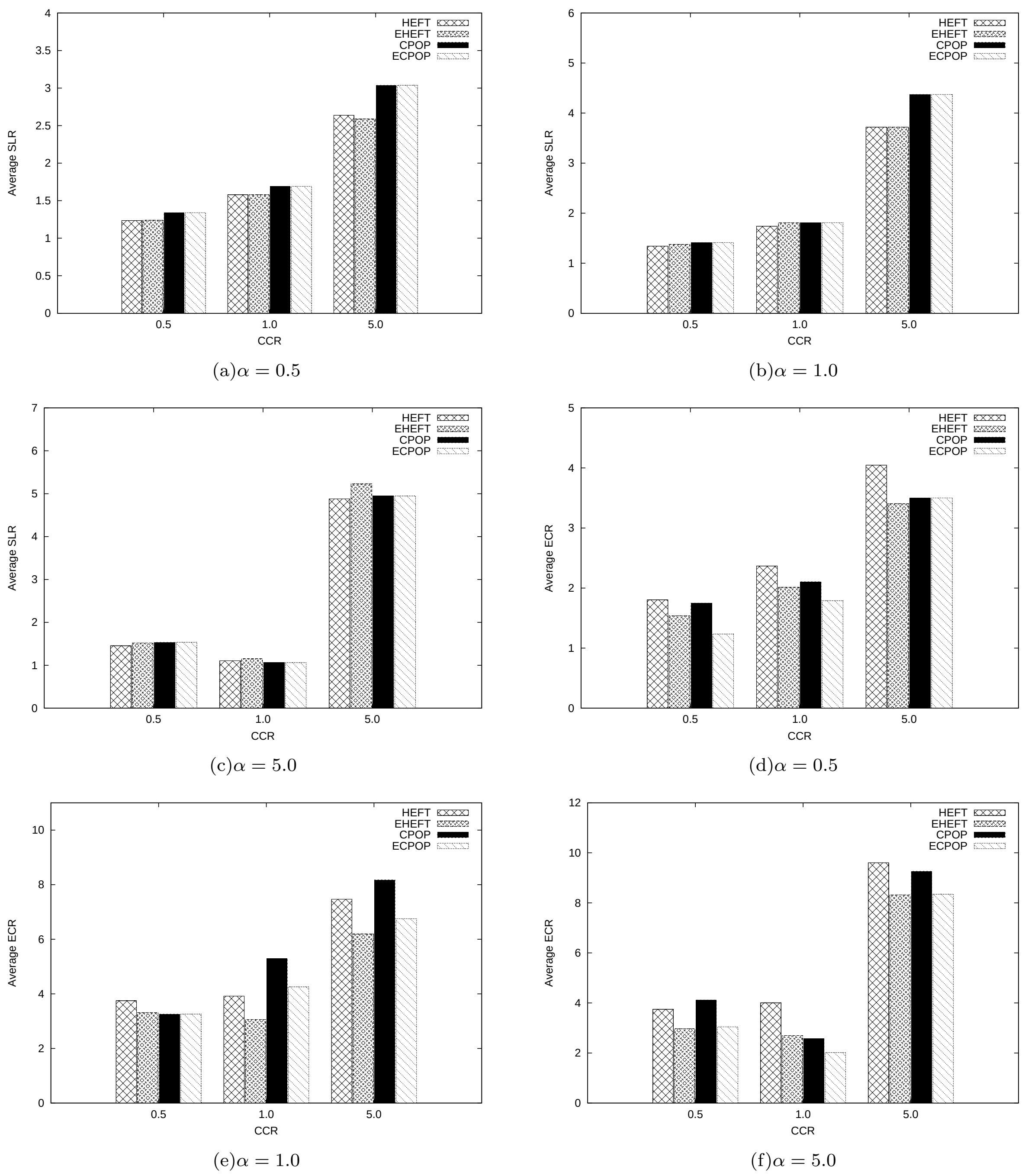
```
Algorithm3 :   Finding a set of inefficient processor
Input :        List of $RE$ for set of $P$ or set of non-critical
Output :       Set of inefficient processor
               Sort $RE$ value
               $P' \longleftarrow \phi$
               $\varepsilon \longleftarrow \frac{1}{\rho}$
               for $\forall\ p_j \in P$ do
                 if $RE_j < \varepsilon$ and $\Sigma_{p' \in P'} RE_{P'} \leqslant \varepsilon$ then
                   add processor $p_j$ to set of
                   inefficient processor
```

4.3.3 Phase 3: Rescheduling

Let P_{nCP} be the new assigned processors excluding the lowest RE processor p_j^{nCP}. Thus,P'_{nCP} is given by $P'_{nCP} = P_{nCP} - p_j^{lowest}$. In this phase, ECPOP uses CPOP algorithm to reschedule a set of non-critical tasks on this new set of processor P'_{nCP} . For example, as illustrated in Fig. 2(d), ECPOP is applied to rearrange the given task in Fig. 2(b) which is scheduled with the CPOP algorithm. In this example, set of processors which is selected as the critical processor P_{nCP} is $\{p_0, p_1, p_2, p_4\}$ and set of non-critical path processor P_{nCP} is $\{p_3\}$. After the RE value of set P_{nCP} is evaluated, the scheduling then selected the processor p_3 (it is the lowest $RE\, p_j^{lowest}$ processor) as the target to shutdown. In this case, we found that the ECPOP does not only reduce the energy consumption from shutting down inefficient processors, but also maintain the makespan of task execution.

5. RESULTS AND DISCUSSION

This section presents the evaluation of our proposed algorithms. The results are obtained from the comparison of EHEFT, ECPOP, HEFT and CPOP algorithm. The metric are schedule length and energy consumption. In this study, the performance is studied using both synthesis task graphs and data taking from real world applications. The simulation parameters are given as follows. For the number of tasks in the synthesis DAG is $\{10, 20, 40\}$. For DAG workload on tasks, we assign the *communication to computation ratio (CCR)* by the set $\{0.5, 1.0, 5.0\}$ (if CCR value is very high, i.e. 5.0, it can be considered as a communication intensive application.). The α is used to depict a shape of the graph. The values of α in this simulation are 0.5 and 1.0. If α value is high, it generates high degree of concurrency (high parallelism application). For heterogeneity factor for processor speed (β) , we choose the set $\{0.5, 1.0\}$ as a range of computation cost (The computation cost among processors is significantly different, if is high.). The set of the number of processors that is available to schedule is $\{5, 10, 15, 20, 25, 30\}$. In the experiments, the number of total task graph evaluated is around 1000 DAGs and three applications on cloud are evaluated extensively. The performance of algorithms



(a)$\alpha = 0.5$ (b)$\alpha = 1.0$

(c)$\alpha = 5.0$ (d)$\alpha = 0.5$

(e)$\alpha = 1.0$ (f)$\alpha = 5.0$

Fig.3: *Average SLR and ECR with* $\alpha = 0.5, 1.0, 5.0$

is evaluated using two metrics: First, the *Schedule Length Ratio (SLR)* which is obtained by dividing the makespan by the summation of the minimum computation cost of task on the critical path. This is given in 3.

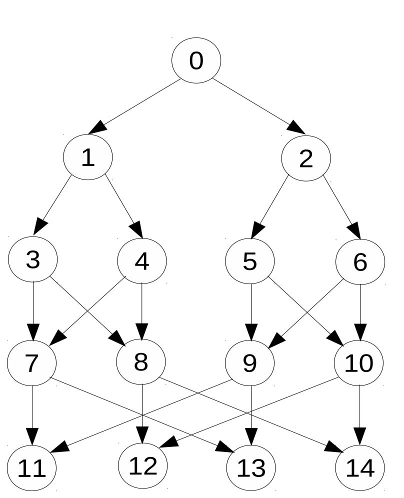
$$SLR = \frac{Makespan}{\Sigma_{t_i \in CP} min_{p_i \in P}\{w_{ij}\}} \tag{3}$$

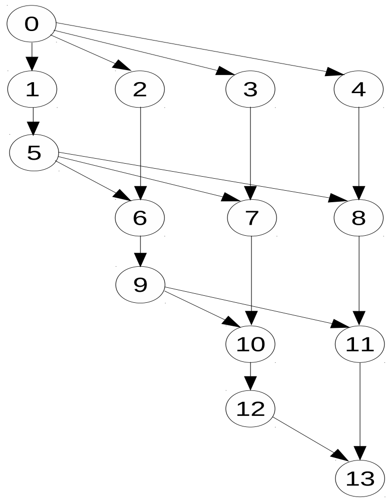
Second, the *Energy Consumption Ration (ECR)* which is obtained by dividing energy consumption by the summation of the energy consumption of tasks, the ECR is given in (4).

$$ECR = \frac{E_{total}}{\Sigma_{t_i \in CP} min_{p_i \in P}\{w_{ij}\} * max_{v_{jk} \in V_j}\{v_{jk}\}^2} \tag{4}$$

5.1 Experimental Results

In this subsection, we present the performance of two proposed algorithms obtaining from the simulation. First, the synthesis task graphs (randomly generated) are used. Then, the data from three real world applications are used.


(a)Fast Fourier Transformation with 4 points


(b)Gaussian Elimination for matrix size 5

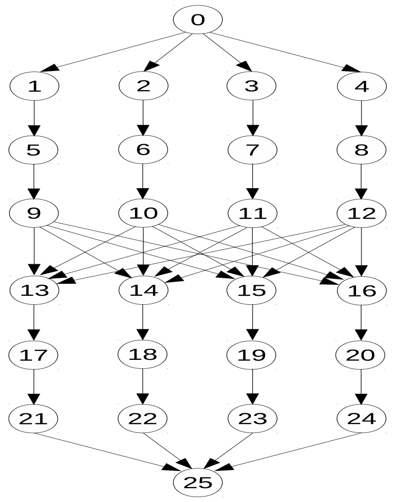

(c)Parallel FI-Growth with 4 input data

Fig.4: *Sample task graphs*

5.1.1 Synthesis Task Graphs

From the result in Fig. 3 and Table 5, it can be seen when EHEFT is employed on the task graph with low degree of parallelism, i.e. $\alpha < 1.0$ and low CCR, the algorithm gives a similar SLR values to HEFT algorithm. However, EHEFT clearly gives a better performance for computation and communication intensive application as shown in Fig. 3(a) and 3(b). For the applications with low degree of parallelism and computation, ECPOP gives an average SLR that is lower than EHEFT and HEFT. However, it can better maintain the application makespan for different type of applications. When considering the value of ECR from Fig. 3(d), 3(e), and 3(f), it can be seen that EHEFT and ECOP can reduce energy consumption more than HEFT and CPOP for various applications. In case of application with high degree of parallelism and high CCR, the SLR value of EHEFT (Fig. 3(c)) increases, since the HEFT usually try to minimize the communication cost among tasks by scheduling task to lower number of assigned processor than another case. As a result, each of the assigned processor will have a higher utilization. Hence, shutting down a few processors will affect the makespan. On average, the average SLR of both proposed scheduling on communication intensive application can achieve its SLR value of less than EHEFT and HEFT. Nevertheless, EHEFT provide results that have substantially lower ECR value, and obtained more solution that reduce the energy consumption than ECPOP, HEFT and CPOP. This is shown in Fig. 3(c) 3(d), 3(e), and 3(f). The comparative results of generated task graphs show that EHEFT and ECPOP can maintain the quality of schedule as similar to HEFT and CPOP. It is not over then 3 % for EHEFT (as shown in Table 5). Clearly, the proposed algorithms can reduce the energy as much as 34 %.

Table 5: *Comparative results of synthesis task graphs*

α	EHEFT		ECPOP	
	SLR	ECR	SLR	ECR
0.5	2.37	17.47	0	11.75
1.0	2.64	34.89	0	25.57
5.0	2.85	18.90	0	17.71

5.1.2 Real World Application Task Graphs

In this work, the application graphs from three real world applications: Fast Fourier Transformation [38], Gauss elimination [25][9] and parallel FI-Growth [37] is also used to evaluate the performance of the proposed algorithm. For Fast Fourier Transformation (FFT), Gaussian elimination (GE), and parallel FI-Growth application, we use the task graph in Fig. 4(a), 4(b) and 4(c) as an input to the simulation. Since the structure of task graph is known, we can ignore some simulation parameters, i.e. number of tasks and shape of task graph (α) . However, communication to computation ratio (CCR) and heterogeneity factor of processor speed (β) are thoroughly applied to conduct various different characteristics of application for more comprehensive experiments. The results are as illustrated in Fig. 5 and Table 6.

Table 6: *Comparative results of three real world application task graphs*

Set of task graph	EHEFT		ECPOP	
	SLR	ECR	SLR	ECR
FFT	2.16	15.69	0	16.89
GE	1.01	13.30	0	17.38
FI-Growth	3.17	6.88	0	7.57

From Fig. 5(a), 5(b) and 5(c), EHEFT and ECPOP algorithm can maintain the same average

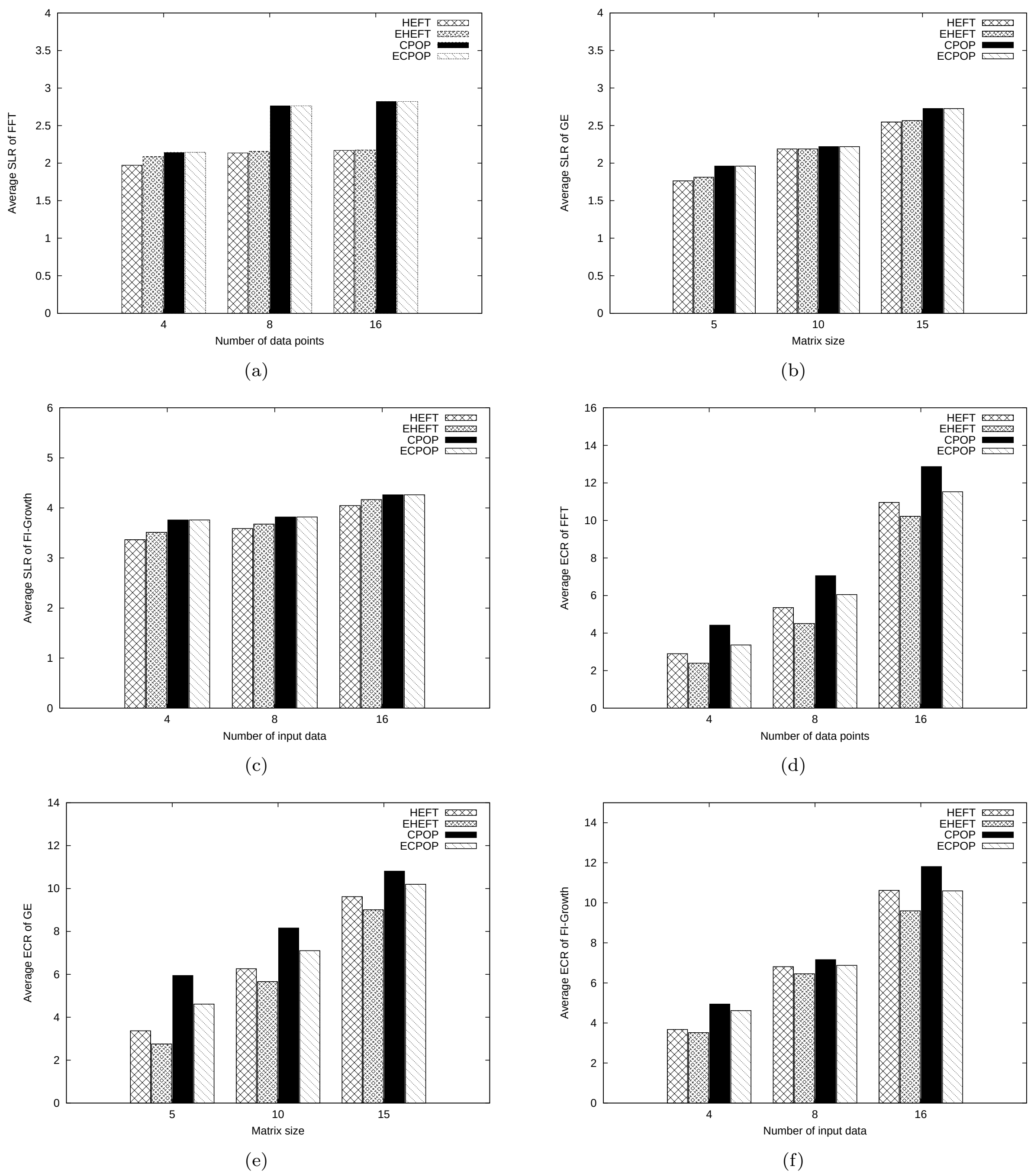


Fig.5: *Average SLR and ECR of FFT, GE and FI-Growth*

SLR value as HEFT algorithm for FFT, GE, and FI-Growth applications. From Table 6, and Fig. 5(d), 5(e) and 5(f), the average value of ECR obtained from the task graph of FFT, GE, and FI-Growth decreases substantially in comparison to HEFT and CPOP. Therefore, one can see that the proposed algorithms can reduce the energy consumption for these three real world applications. Anyway, the makespan increases slightly due to the attempt to reschedule task for energy saving. In addition, the energy can be saved as much as 19 % but the average makespan increases is less than 8 %. Finally, the results for our study are summarized in Table 6. It can be seen that EHEFT algorithm mostly outperformed HEFT with communication intensive application. Anyway, EHEFT result in a makespan increases for computation intensive application. For EHEFT, ECR value decreases substantially compared to the three others scheduling algorithms (ECPOP, HEFT, and CPOP). However, EHEFT does not maintain the scheduling makespan for compute intensive application. Thus, ECPOP is a better choice when the preservation of the makespan is needed.

For the complexity of the proposed algorithms, the

analysis is as follows. For a task graph with T tasks and P processors:
1. First, HEFT or CPOP is applied, the complexity of the algorithm is $T_1 = \mathcal{O}(\mathcal{T} \times \mathcal{P})$.
2. The RE metric is used to evaluate all processors, this is process is $T_2 = \mathcal{O}(\mathcal{P})$.
3. Task graph is rescheduled again on a selected set of processors using either HEFT or CPOP. Thus, the complexity of this phase is the same as the first phase $T_3 = \mathcal{O}(\mathcal{T} \times \mathcal{P})$.

From these results, the complexity of both EHEFT and ECPOP is given by $T_{total} = \mathcal{O}(\mathcal{T} \times \mathcal{P}) + \mathcal{O}(\mathcal{P}) + \mathcal{O}(\mathcal{T} \times \mathcal{P})$ or $T_{total} = \mathcal{O}(\mathcal{T} \times \mathcal{P})$. The complexity of these proposed algorithms depends on the number of tasks and processors used. As the number of task increased, the time spent will depends more on the number of tasks.

In this work, the energy consumed by a server at boot up or shut down time is not considered. Although, the booting up or shutting down of a computing server can consume some energy, the energy saved by turning off the system is usually greater. The attainable benefit is even greater a long running time task which is the target of this work. If the power consumption at the boot up and shutdown time is consider, the total power consumption of the system will be slightly increased from the analysis. Let n depicts the number of server being turn off according to the proposed algorithms. The booting up process consumes energy equal to E_{boot} and shutting down process consumes energy equal to $E_{shutdown}$. Then, the additional power consumption is given by $n(E_{boot} + E_{shutdown})$. This energy consumption will slightly increase the total energy consumption obtained from the previous analysis. But there is no impact to the key concept of the algorithms proposed.

6. CONCLUSION

In this paper, two energy aware scheduling for the cloud application called *EHEFT* and *ECPOP* are proposed. The goal is to reduce energy consumption while maintaining the performance of scheduling as much as possible. The proposed algorithms use performance metric called **RE** to identify inefficient processors. The first algorithm, EHEFT, focuses on reducing the energy usage but allow the task makespan to increases when necessary. The second algorithm, ECPOP, takes into account the makespan conservation and energy consumption reduction. Thus, the energy reduction is less than EHEFT but the scheduling are more efficient. The simulation results of synthesis and real world task graphs show that the proposed algorithms help reduce the energy consumption substantially. In addition, the experimental results show that EHEFT and ECPOP can reduce the power consumption for many different classes of parallel applications. The result of this work can be applied to reduce the operation cost of a large cloud data center. In the future, improving a quality of scheduling in certain complex applications can be investigate along with how to create a more efficient use of different techniques such as task duplication and task clustering.

References

[1] J. G Koomey, “ Worldwide electricity used in data centers,” *Environmental Research Letters*, Vol.3, No.3, pp.973-994, 2008.

[2] J. G. Koomey, “Estimating total power consumption by servers in the U.S. and the world,” *Technical report, Lawrence Derkley National Laboratory*, 2007.

[3] J. Shuja and S. A. Madani and K. Bilal and K. Hayat and S. U. Khan and S. Sarwar,“Energy-efficient data centers,” *Journal of Computing*, Vol. 94, No. 12, pp.973-994 ,2012.

[4] G. Valentini, W. Lassonde, S. U. Khan, M.-A. Nasro, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, “An overview of energy efficiency techniques in cluster computing systems,” *Journal of Cluster Computing* , Vol.16, No. 1, pp.3-15, 2013.

[5] J. Mei, K. Li, and K. Li, “Energy-aware task scheduling in heterogeneous computing environments, ”*Journal of Cluster Computing* , Vol.17, No.2, pp.537-550, 2014.

[6] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, “A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters,” *Future Generation Computer Systems*, Vol. 37, No.1, pp. 141-147, 2014.

[7] Y.C. Lee, and A.Y. Zomaya, “Energy efficient utilization of resources in cloud computing systems,” *The Journal of Supercomputing*, Vol.60, No.2 , pp.268-280, 2012.

[8] Y. Ma, B. Gong, R. Sugihara, and R. Gupta, “Energy-efficient deadline scheduling for heterogeneous systems,” *Journal of Parallel Distributed Computing*, Vol. 72, No. 12 , pp. 1725-1740, 2012.

[9] M. Cosnard, M. Marrakchi, Y. Robert and D. Trystram,“Parallel Gaussian elimination on an MIMD computer,” *Journal of Parallel Computing*, Vol. 6, No. 3, pp. 275-296, 1988.

[10] T. D. Burd and R. W. Brodersen,“Energy efficient CMOS microprocessor design,” *Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS ’95)*, Jan. 1995, Hawaii, pp.288-297 .

[11] L. Wang,G. v. Laszewski,J. Dayal, F. Wang, “Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS,” *Proceedings of the 2010 The 10th*

IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'10), May. 2010, Melbourne ,pp.368-377.

[12] V. Venkatachalam and M. Franz," Power reduction techniques for microprocessor systems. "*ACM Computing Surveys* , Vol. 37, No. 3, pp.195-237, 2005.

[13] D. Zhu, R. Melhem, and B. R. Childers," Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 7, pp. 686-700, 2003.

[14] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *SIGOPS Operating Systems Review*, Vol. 35, No.5, pp. 89-102, 2001.

[15] K. Hoon Kim, R. Buyya, and J. Kim," Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters." *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID '07)*, May.2007, Rio de Janeiro, pp.541-548.

[16] Q. Huang, Sen Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced Energy-Efficient Scheduling for Parallel Applications in Cloud," *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID '12)*, May. 2012, Washington DC, pp.781-786.

[17] Y. C. Lee and A. Y. Zomaya, "Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions." *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22,No. 8, pp. 1374-1381, 2011.

[18] Y. C. Lee and A. Y. Zomaya," Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling." *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*,May 2009, Shanghai, pp.92-99 .

[19] M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.-G. Talbi, A.Y. Zomaya, and D. Tuyttens," A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing*, Vol. 71, No. 11, pp. 1497-1508, 2011.

[20] F. Gruian, and K. Kuchcinski," LEneS: task scheduling for low-energy systems using variable supply voltage processors," *Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference* ,Feb. 2001,New York ,pp.449-455.

[21] Y. Zhang, X. Hu and D.Z.Chen, "Task scheduling and voltage selection for energy minimization," *Proceedings 39th Design Automation Conference*, Jun. 2002, Louisiana ,pp.183-188 .

[22] X. Zhong, and C.-Z. Xu, "Energy-Aware Modeling and Scheduling for Dynamic Voltage Scaling with Statistical Real-Time Guarantee," *IEEE Transactions on Computers*, Vol.56, No.3, pp.358-372, 2007.

[23] Y.K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys* , Vol. 31, No.4, pp.406-471, 1999.

[24] S. Darbha and D. P. Agrawal," Optimal Scheduling Algorithm for Distributed-Memory Machines, " *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9,No. 1,pp.87-95, 1998.

[25] H. Topcuouglu, S. Hariri, and M.Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3 , pp.260-274, 2002.

[26] T. D. Braun, H.J. Siegel, N. Beck, L. L. Blni, M. Maheswaran, A.I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund,"A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, Vol. 61,No. 6 , pp. 810-837, 2001.

[27] P.F. Dutot, Tchimou N'Takpe, F. Suter, and H. Casanova," Scheduling Parallel Task Graphs on (Almost) Homogeneous Multicluster Platforms," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 20,No. 7, pp. 940-952, 2009.

[28] Z. Shi and J. J. Dongarra, "Scheduling workflow applications on processors with different capabilities," *Future Generation Computer Systems*, Vol. 22, No. 6, pp.665-675, 2006.

[29] A. Y. Zomaya, C. Ward, and B. Macey, "Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues," *IEEE Transactions on Parallel and Distributed Systems*, Vol.10,no. 8 , pp.795-812, 1999.

[30] M. Sharifi, S. Shahrivari, and H. Salimi, "PASTA: a power-aware solution to scheduling of precedence-constrained tasks on heterogeneous computing resources," *Journal of Computing*, Vol.95, No. 1 , pp. 67-88, 2013.

[31] A. Salman, I. Ahmad, A.-M. Sabah, "Particle swarm optimization for task assignment problem," *Journal of Microprocessors and Microsystems*, Vol. 26, No. 8, pp. 363-371, 2002.

[32] Z. Mousavinasab, E.-M. Reza and A. Movaghar ,"A Bee Colony Task Scheduling Algorithm in Computational Grids," *Proceedings of International Conference ICDIPC 2011*, Jul. 2011,Ostrava, pp.200-210.

[33] R. Deng, C. Jiang, and F. Yin, "Ant colony optimization for precedence-constrained heterogeneous multiprocessor assignment problem,"

Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09), 2009, New York, pp. 89-96.

[34] S. Baskiyar and A.-K. Rabab , "Energy aware DAG scheduling on heterogeneous systems," *Journal of Cluster Computing*, Vol. 13, No. 4 ,pp. 373-383, 2010.

[35] J.-C. Liou and M. A. Palis,"An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors," *Proceedings of The 8th IEEE Symposium On Parallel And Distributed Processing* ,Oct. 1996, Louisiana, pp.152-156.

[36] J. Zhuo and C. Chakrabarti,"Energy-efficient dynamic task scheduling algorithms for DVS systems," *ACM Transaction on Embedded Computing*, Vol.7, No. 2, pp.1-25, 2008.

[37] N. Benjamas and P. Uthayopas, "Enhancing Parallel Data Mining Performance on a Large Cluster by Using UCE Scheduling," *JNIT: Journal of Next Generation Information Technology*, Vol. 2, No. 4, pp. 69 - 77, 2011.

[38] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms* , MIT Press, 2009, Massachusetts, ch.30.

Thanawut Thanavanich received his bachelor in computer science from Ramkhamhaeng University in 2001 and received master degree in computer engineering from Kasetsart University in 2005. He is currently a Ph.D student in computer engineering at Kasetsart University and a lecturer of computer engineering program at Chiang Rai Rajabhat University. His main areas of research interests include parallel computing and distributed computing.

Putchong Uthayopas received his bachelor and master degree in electrical engineering from Chulalongkorn University in 1984 and 1988. He received master and PhD in computer engineering from University of Louisiana in 1994 and 1996 accordingly. His research interest is in cluster computing, grid and cloud computing system and tools. He published more than 130 refereed publication in conferences and Journals. Putchong Uthayopas is a co-founder of the Thai National Grid project. In 2012, he received a distinguish computer engineer award in system integration from the Engineering Institute of Thailand.