Cost-Aware Scheduling of Computation-Intensive Tasks on Multi-Core Server

Youwei Ding[1], Liang Liu[2*], Kongfa Hu[1] and Caiyan Dai[1]
[1] College of Information Technology, Nanjing University of Chinese Medicine
Nanjing, 210023 - China
[e-mail: dingyouwei@nuaa.edu.cn, kfhu@njutcm.edu.cn, daicaiyan@163.com]
[2] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
Nanjing, 210016 - China
[e-mail: liangliu@nuaa.edu.cn]
*Corresponding author: Liang Liu

Received October 14, 2017; revised December 22, 2017; accepted May 8, 2018; published November 30, 2018

Abstract

Energy-efficient task scheduling on multi-core server is a fundamental issue in green cloud computing. Multi-core processors are widely used in mobile devices, personal computers, and servers. Existing energy efficient task scheduling methods chiefly focus on reducing the energy consumption of the processor itself, and assume that the cores of the processor are controlled independently. However, the cores of some processors in the market are divided into several voltage islands, in each of which the cores must operate on the same status, and the cost of the server includes not only energy cost of the processor but also the energy of other components of the server and the cost of user waiting time. In this paper, we propose a cost-aware scheduling algorithm ICAS for computation intensive tasks on multi-core server. Tasks are first allocated to cores, and optimal frequency of each core is computed, and the frequency of each voltage island is finally determined. The experiments' results show the cost of ICAS is much lower than the existing method.

Keywords: Temporal cost, energy cost, task scheduling, multi-core server

This research was supported by the National Natural Science Foundation of China (No. 81674099, No. 61402225), Qing Lan Project of Jiangsu Province of China (2016), National Natural Science Foundation of Jiangsu (BK20140832) and a project funded by Priority Academic Program Development of Jiangsu Higher Education Institutions(PAPD).

http://doi.org/10.3837/tiis.2018.11.016 **ISSN : 1976-7277**

1. Introduction

Cloud computing is now becoming the prominent new paradigm in distributed systems. It takes advantage of the economies of scale, and advanced services can be provided to users in a pay-as-you-go manner. Cloud is cost effective for users, but it is a challenge for the service providers to power the increasing hardware resources, because the cost of energy consumed by a server during its lifetime has exceeded the cost of the server itself [1]. The server is the physical unit of the cloud platform, and the servers and storage devices consume about 26% of the total energy of the data center [2]. The world's 44 million servers consumed 0.5 percent of total electricity in 2008, and it will be quadruple by 2020 if the demand continues [3]. However, the utilization of a typical data center is around 20-30% [4], which means a large amount of energy will be wasted. Meanwhile, the environment impact is another challenge for data centers since the majority of today's energy is generated from non-renewable fossil fuels [5], which produce harmful CO_2 emissions. According to the Smart 2020 report [6], data centers will be responsible for 18 % CO_2 of emission in the world. Hence reducing the energy consumption of the servers can greatly profit the services providers, users and the environment.

The servers typically have one or more multi-core processors nowadays. The cores of the processor are divided into several voltage islands (or groups), all the cores in the same island operate on the same frequency, while the operating frequencies of the core in different islands are independent. Energy is consumed when processing tasks on multi-core server, and the users should wait for the results of the tasks. In cloud computing, users rent the resources (or server) in a pay as you go manner to run their jobs. If the energy consumption of the rented server is high, the profit of the service provider becomes less. And if the processing time of the job is too long, the probability that the user gives up using the service grows greater. Hence both the energy consumption and the waiting time of the user should be taken into consideration when processing jobs on multi-core server, especially in cloud computing environment.

Most existing methods only focus on the energy consumed by processor, however the energy consumed by the other components cannot be ignored when we improve the energy efficiency of the multi-core server. And the existing research aim to improve either the energy efficiency or performance, these methods can reduce the economic cost or improve the quality of service for the users. But both the energy cost and the time cost should be reduced simultaneously for the service provider to gain higher profit. A cost-aware scheduling method was proposed in [7], but it only suits for the processor in which each core operates on independent frequency.

In this paper, we define the cost of processing a computation-intensive task on the multi-core server in which the cores of the processor are divided into several voltage islands, and propose a cost-aware scheduling algorithm ICAS for the multi-core server to reduce the cost when processing computation-intensive tasks. In ICAS, the processor is fully used and the workload of each core is as balanced as possible to reduce the energy waste and waiting time. The scheduling is divided into three steps, allocating tasks to cores, computing the optimal frequency of each core, and determining the frequency of each voltage island. The experiments' results show the cost of ICAS is much lower than the existing method.

The rest of the paper is organized as follows. We overview the related works in section 2, and introduce the preliminaries of the issue in section 3. In section 4, we introduce how to

schedule the tasks on a single core server. And we describe the details of the proposed algorithm in section 5, and show the experiments' results in section 6. We conclude our work in section 7.

2. Related Work

Multi-core processors have been widely used in servers, PCs and mobile phones. Many researches aim to reduce the power of the multi-core processor, and improve the energy efficiency of processing tasks on multi-core processors. DVFS, which changes the operating frequency of the cores, and DPM (Dynamic Power Management), which chooses proper cores to run the tasks, are two common techniques to reduce the energy consumption of the multi-core processor.

Liu et al proposed an energy-aware method EA-DVFS to select the voltage and the frequency of the processor in embedded systems in [8]. It changes the status of the processor according to the remainder energy, the processor operates on max frequency if there are enough energy, otherwise, lower frequency will be used to save energy. A variable-aware DVFS method was proposed in [9], in which the status of the processor was changed according to the variables of voltage, temperature, process parameter and so on, rather than using frequency threshold and greedy policy. In [10] the models were made for both homogeneous and heterogenous processors, the authors tried to reduce the dynamic power of the processor, and compared the dynamic energy consumption for processing tasks on the two kinds of processors. To find the optimal frequency of each core when processing tasks, decision tree was adopted in [11] to minimize the energy consumption of each user instruction. However, only the energy of the multi-core processor is considered rather than the energy consumption of the whole server in these methods.

The energy efficient scheduling methods based on DVFS only consider the dynamic power of the processor, which is proportional to the square of the operating frequency. Energy efficient scheduling of multiple tasks on multi-core processor was proved to be an NP problem in [12]. This issue can be deduced to number partition problem, which is the simplest NP problem. But the solution of number partition cannot work well when there are more than two partitions [13]. The scheduling can also be deduced to other classic NP problems such as bin packing, integer linear programming and so on.

With the development of the nanofabrication technology of the processor, the static power is becoming more and more important in the total power, and only reducing the dynamic power cannot decrease the power of the processor in some cases. Hence more and more researches integrated DVFS and DPM techniques to schedule the tasks energy efficiently on multi-core servers. However, these researches assumed that the cores are controlled independently, i.e. the cores can operate on different frequencies. This cannot suit for majority of processors on the market, because the cores of the most multi-core processors are divided into several voltage islands, and the cores in the same island must operate on the same frequency.

Kong et al scheduled the real-time tasks on the multi-core processor with multiple islands in [14]. The optimal frequency of the island was determined by the number of cores and the static power when processing tasks without deadline. A method with polynomial complexity was proposed to compute the minimal energy for processing tasks with deadline constraint. The method first computed the number of islands to run tasks, assigned the tasks to each core, and find the optimal frequency of each island. SFA (Single Frequency Approximation) was

used in [15], all the cores of each island operate on the same frequency when processing tasks. A dynamic programming method was adopted to allocate the tasks and set the frequencies, according to the approximation factor of energy consumption in SFA. The energy efficiency of SFA was analyzed in [16], it is found that the energy efficiency of SFA decreases when there were more cores in a voltage island. Another conclusion was that SFA is very practical and can be integrated with other dynamic power management techniques to improve the energy efficiency, though the schedule was not always the optimal solution. The most workload first was proposed in [17], and tasks were first allocated to islands with most workload to improve the resource utilization of the islands and then reduce the energy consumption. And the method was theoretically analyzed, and the approximation ratio was given.

As to the server with multi-core processor, the energy model for scheduling the tasks is more complex. Three measurements were defined for selecting energy efficient schedules of the tasks processing on multi-core server in [18]. The measurements were determined by the idle and peak power of the server and the parallelism of the tasks. A power and energy container was designed in the level of operating system in [19] to compute and control the power and energy usage of each fine-grained request. Tseng et al tested the energy of a server in [20], and found that scheduling the multi-thread programs properly can reduce the power cost and energy consumption of the server. These researches developed some methods to reduce the energy or improve the energy efficiency of the server, and the cores of the server are assumed to be controlled independently.

There are still some work take both energy consumption and performance into consideration. A GE scheduling algorithm was proposed in [21] to tradeoff the energy consumption of the server and QoS of the applications. The core speed was transformed to compensate the service quality in GE. Workload dependent dynamic power management was proposed in [22] to reduce the energy and improve the performance of the server. However, these methods are not practical for the service providers of cloud computing to manage the server resources, because the service is provided in a “pay as you go” manner. Thus the scheduling solutions must be cost aware when processing tasks on the multi-core server.

A cost aware scheduling algorithm was proposed in [7], the cost of the server for processing a task contains the energy cost and temporal cost. Two task execution modes, the batch mode and the online mode, are considered, and heuristics are developed. This work is similar to our paper, but an assumption was made in this wok that all the cores of the server can be controlled independently. Hence, the method is not suit for situation in this paper that the cores are divided into several voltage islands.

3. Preliminaries

A physical node *PM* consists of processor, disk, memory, mainboard and other components, and the processor is DVFS supported. It is supposed that the power of the processor, P_{cpu}, varies with its operating frequency while that of the other components, P_s, remains constant when *PM* process computation-intensive tasks. We also suppose that only one processor is equipped on *PM*, and no overhead of power and time will be cost when the processor changes its operating frequency. Then the energy consumed by *PM* is $E = \int_{st}^{et} \left(P_{cpu}(t) + P_s \right) dt$, where *st* and *et* are the start and end time for processing the task, and $P_{cpu}(t)$ is the power of the processor at time *t*.

The power of the processor P_{cpu} includes dynamic power P_d and static power (or leakage power) P_l. The dynamic power can be formed as $P_d = \alpha \cdot f^h$, where α and h are constants and h=3 for simplicity. It is assumed that static power is independent of the operating frequency in most cases. Suppose $P_l = \beta \cdot P_{d,f_{\max}}$ and $P_s = \gamma \cdot P_{d,f_{\max}}$, where $P_{d,f_{\max}}$ is the maximum dynamic power, β and γ are constants. Then we get the energy consumption of *PM* is

$$E = P_{d,f_{\max}} \cdot \int_{st}^{et} \left(f^h(t) + \beta + \gamma \right) dt \tag{1}$$

A computation-intensive task $J_i = (r_i, d_i, e_i)$, where r_i and d_i are the release time and deadline of the task, and e_i is the worst case execution time (WCET) which is the execution time for the processor operating on maximum frequency f_{max} to complete J_i. We normalize the maximum frequency f_{max}=1, then the execution time of J_i with frequency f is $t_{J_i,f} = f \cdot e_i$ and the energy consumption of the node is

$$E_{J_i,f} = \left(P_s + P_l + P_{d,f} \right) \cdot t_{i,f} = \left(\beta + \gamma \right) \cdot P_{d,f_{\max}} \cdot f^{-1} \cdot e_i + f^2 \cdot P_{d,f_{\max}} \cdot e_i \tag{2}$$

The cost of a node for processing the task includes temporal cost C_t and energy cost C_e, i.e. $Cost=Cost_t+Cost_e$. Temporal cost is the amount of money paid to compensate a user for waiting for his/her task to be completed, $Cost_t = R_t \cdot t$, where R_t is the temporal factor and t is the execution time of the task. Energy cost represents the money paid for the energy consumption for processing the task, $Cost_e = R_e \cdot E$, where R_e is energy factor which can be derived from the electricity price, E is the energy consumption of the server.

Tasks may arrive together without deadline constraint, which is called batch mode, or arrive unpredictable and each task has an explicit deadline constraint, which is called online mode. In this paper we focus on scheduling the tasks in batch mode on a server whose cores are consisting of multiple voltage islands, which are also called groups or clusters, so that the total cost for processing the tasks is minimal. To solve this problem, we first schedule batch tasks on a single core server.

4. Cost-Aware Scheduling on Single Core Server

In the batch mode, a set of tasks $J = \{J_1, J_2, \cdots, J_n\}$ without deadline constraints arrive simultaneously and can be processed in arbitrary order. If the tasks are sequentially processed and the processor has single core, the cost of the node for processing task J_i is

$$Cost_{J_i} = Cost_{t,J_i} + Cost_{e,J_i} = R_t \cdot \sum_{j=1}^{i} t_j + R_e \cdot E_{J_i} \tag{3}$$

where $Cost_{t,J_i}, Cost_{e,J_i}$ are the temporal cost and energy cost of the node for processing J_i, t_i is the time duration between the start and end of the execution of J_i, E_{J_i} is the energy consumption for processing J_i. Then the time for waiting for J_i to be completed is $\sum_{j=1}^{i} t_j$.

It can be seen that the temporal cost for each task is caused by the execution of itself and the tasks processed ever before. And the total cost of the tasks in J is

$$\begin{aligned} Cost &= \sum_{J_i \in J} Cost_{t,J_i} + Cost_{e,J_i} = R_t \cdot \sum_{J_i \in J} \sum_{j=1}^{i} t_j + R_e \cdot \sum_{J_i \in J} E_{J_i} \\ &= R_t \cdot \sum_{J_i \in J} (n - i + 1) \cdot t_i + R_e \cdot \sum_{J_i \in J} E_{J_i} \end{aligned} \tag{4}$$

where R_t and R_e are the temporal and energy factors, n is the number of tasks, and i is the execution order of task J_i.

In other words, the temporal cost for J_i can also be viewed as the sum of the temporal cost for itself and the left tasks caused by the execution of J_i. Then the cost for processing J_i will be

$$\begin{aligned} Cost_{J_i} &= (n-i+1)\cdot R_t \cdot t_i + R_e \cdot E_{J_i} \\ &= (n-i+1)\cdot R_t \cdot e_i / f + R_e \cdot (\beta+\gamma)\cdot P_{d,f_{\max}} \cdot e_i / f + R_e \cdot f^2 \cdot P_{d,f_{\max}} \cdot e_i \end{aligned} \tag{5}$$

where e_i is the WCET of J_i, and f is the operating frequency of the processor for processing J_i.

To minimize the total cost, we should consider the following two issues, how to determine the execution order of the tasks and how to choose optimal frequency for the server to process each task. It is shown in Eq.(5) that the tasks executed earlier may lead to more temporal cost, because the execution time of the task affect the temporal cost of the succedent tasks.

Theorem 1. Executing the batch tasks according to their WCET in non-decreasing order can minimize the cost of the single core server, if the server operates on the same frequency.

Proof. Since the server operates on the same frequency, the energy cost of each task will not change in different execution order. Given the set of tasks $J=\{J_1, J_2, \cdots, J_n\}$ in the batch mode, and the WCET of the tasks $e_1 \le e_2 \le \cdots \le e_n$. The temporal cost is CT_1 when the tasks are sequentially executed. If the tasks J_i and J_j ($1 \le i < j \le n$) are swapped, the temporal cost for the tasks is CT_2. Then we have

$$\begin{aligned} CT_2 - CT_1 &= (n-i+1)\cdot R_t \cdot (e_j - e_i)/f - (n-j+1)\cdot R_t \cdot (e_j - e_i)/f \\ &= R_t \cdot (j-i)\cdot (e_j - e_i)/f \end{aligned} \tag{6}$$

It is obvious that $CT_2 \ge CT_1$, as $j>i$ and $e_j \ge e_i$. Therefore, executing the batch tasks according to their WCET in non-decreasing order leads to minimal cost.

It can also be seen that the only variable in Eq.(5) is f_i when the execution of task J_i is given. Then the derivation of $Cost_{J_i}$ is computed and set to be 0, we can get the theoretical frequency f^* to minimize the cost

$$f^* = \sqrt[3]{\frac{(n-i+1)\cdot R_t}{2R_e \cdot P_{d,f_{\max}}} + \frac{\beta+\gamma}{2}} \tag{7}$$

Since the operating frequencies of each island are discrete, that is the processor cannot operate on theoretical frequency in most cases. The processor should operate on the optimal frequency, which is also called practical frequency, when processing task J_i to minimize the cost. Thus we have

$$f^{opt} = \begin{cases} f_{\max}, & f^* > f_{\max} \\ f_{\min}, & f^* < f_{\min} \\ f_i^*, & f^* \in F \\ \arg\min\{Cost(f_i), Cost(f_{i+1})\}, & f_i < f^* < f_{i+1} \end{cases} \tag{8}$$

where $Cost(f_i)$ and $Cost(f_{i+1})$ are the cost of the server for processing task J_i when operating on frequencies f_i and f_{i+1} respectively. It can be seen that the optimal frequency for processing task J_i only depends on the execution order of the task. If the task J_i is executed before J_j ($i<j$), we have $f_i' \ge f_j'$ according to Eq.(7) and Eq.(8).

Therefore, cost-aware processing of batch tasks on single core server should be divided into two steps. The tasks should be firstly sorted according to their WCET in non-decreasing order. Then the server operates on optimal frequency of each task computed by Eq.(7) and Eq.(8) to execute the corresponding task to minimize the cost.

5. Cost-Aware Scheduling on Multi-Core Server

On the multi-core server, the processor consists of m homogeneous cores, and the set of operating frequencies of the processor is $F=\{f_1, f_2, ..., f_{ns}\}$, where ns is the number of frequencies the processor can operate on. We assume $f_1< f_2< ...< f_{ns}$ without loss of generality, then the minimum frequency $f_{min}=f_1$ and the maximum frequency $f_{max}=f_{ns}$. The power of the processor is $P_{cpu}=\sum_{i=1}^{m}\left(P_{i,d}+P_{i,l}\right)$, where $P_{i,d}$ and $P_{i,l}$ are the dynamic power and static power of each core C_i ($1\le i\le m$), and $P_{i,l}=P_l/m$, $P_{i,d}(f)=P_{d,f}/m$, $P_{i,d}(f)$ is the dynamic power of C_i when it operates on frequency f.

We also assume that task cannot be processed on more than one cores at the same time, but it can be migrated from one core to another. If all the cores are idle, the processor and the node will be transmitted into low power state, and the power of the node is 0. Otherwise, the processor and the node are active.

The cores of the processor are divided into some groups (also called voltage islands). The cores in a voltage island operate on the same frequency, while the voltage islands operate on various frequencies. In other words, the unit of the cores for changing operating status is the voltage island. Each voltage island typically includes the same number of cores. Therefore, the power of the node will be

$$P=P_s+\sum_{i=1}^{M}\left(x(i)\cdot\left(P_{M_i,d}\left(f_{M_i}\right)+P_{M_i,l}\right)\right)=P_s+\sum_{i=1}^{M}\left(x(i)\cdot\frac{m}{M}\cdot\left(P_{C_{i,j},d}\left(f_{M_i}\right)+P_{C_{i,j},l}\right)\right)$$

where m and M are the number of the cores and the voltage islands, each voltage island contains m/M cores, M_i is the i-th group of the processor, $C_{i,j}$ is the j-th core in M_i, f_{M_i} is the operating frequency of the cores in M_i, $P_{M_i,d}\left(f_{M_i}\right)$ and $P_{M_i,l}$ are the dynamic and static power of M_i, $P_{C_{ij},d}\left(f_{M_i}\right)$ and $P_{C_{ij},l}$ are the dynamic and static power of core $C_{i,j}$. If M_i is active, $x(i)=1$, otherwise, $x(i)=0$.

Since all the cores in a voltage island M_i operate on the same frequency, then the optimal frequency of the island will be $f_{M_i}=\max_{C_{i,j}\in M_i}\left\{f_{i,j}^{opt}\right\}$, where $f_{i,j}^{opt}$ is the optimal frequency of the core $C_{i,j}$ on island M_i. The workload $W_{i,j}$ of each core $C_{i,j}$ is the sum of WCET of the tasks on $C_{i,j}$, i.e $W_{i,j}=\sum_{J_s\in Q_{i,j}}e_s$, where $Q_{i,j}$ is the set of tasks allocated on $C_{i,j}$. The workload W_i of each voltage island M_i will be $W_i=\max_{C_{i,j}\in M_i}\left\{W_{i,j}\right\}$. When a task J_a is assigned to the core $C_{i,j}$ on island M_i, the increamental workload of M_i is

$$\Delta W_i=\begin{cases}0, & e_a+W_{i,j}\le W_i\\ e_a+W_{i,j}-W_i, & otherwise\end{cases} \tag{9}$$

where e_a is the WCET of task J_a. Hence, the workload of M_i will be $W_i'=W_i+\Delta W_i$, after assigning task J_a is assigned to core $C_{i,j}$ on M_i.

Though the operating frequency of each voltage island can be transformed, three issues should be taken into consideration for processing the tasks energy efficiently. ① Which islands are used to process the tasks? ② How to determine the execution order of the tasks on each core. ③ How to determine the frequency of each selected island.

Once the tasks are allocated to each core, the execution order of the tasks and operating frequency of each core can be determined according to Theorem 1 and Eq.(8). Therefore, the

key issue is tasks allocation on multi-core server to minimize the cost of the server. The resource of cores in each voltage islands should be fully utilized to reduce the energy wastes, and more islands will be used to balance the workload of the cores to reduce the waiting time of the tasks.

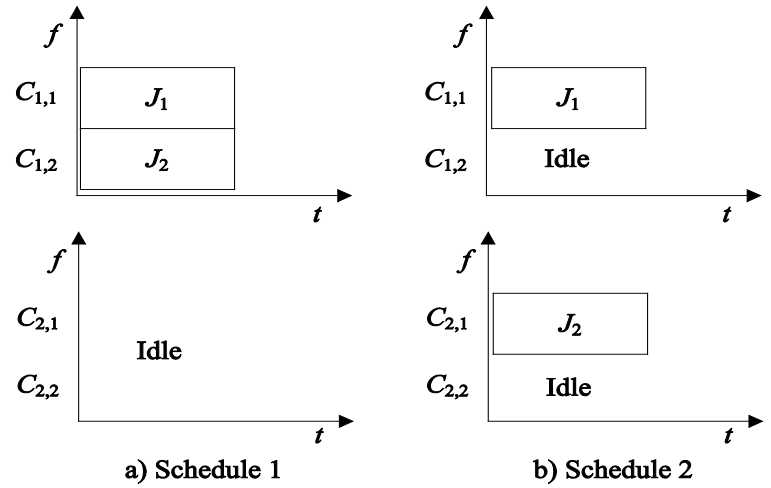


Fig. 1. Two tasks processed on a server

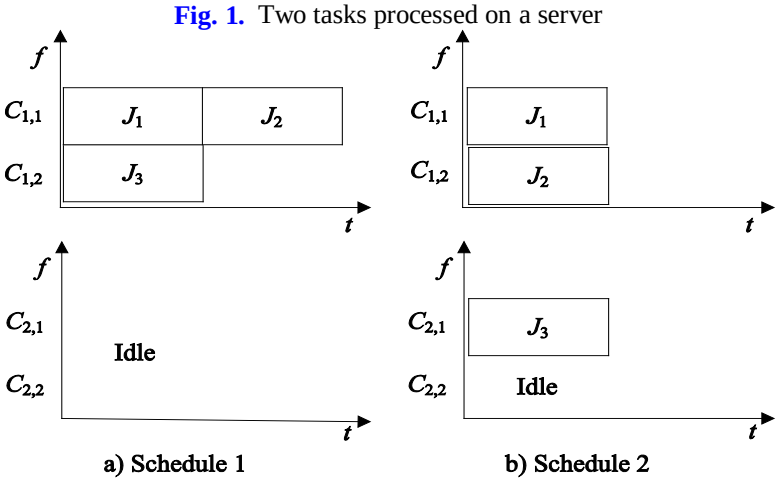


Fig. 2. Three tasks processed on a server

Example Given a server *PM* whose maximum dynamic power, static power and power of other components are $P_{d,f_{\max}} = P_l = P_s$. The processor of *PM* has two voltage islands M_1 and M_2, each of them has two cores $C_{1,1}$ and $C_{1,2}$, and $C_{2,1}$ and $C_{2,2}$ respectively. Two sets of tasks $J=\{J_1, J_2\}$, $J'=\{J_1, J_2, J_3\}$ are needed to be processed, and WCET of the tasks are $e_1 = e_2 = e_3 = e$.

When the set of tasks J is processed, two schedules can be made as shown in **Fig. 1**. In **Fig. 1 a)**, tasks J_1 and J_2 are executed on core $C_{1,1}$ and $C_{1,2}$ respectively, and cores $C_{2,1}$ and $C_{2,2}$ are idle. The cost of the server and the theoretical frequency of core $C_{1,1}$ and $C_{1,2}$ will be $Cost_a = 2 \cdot \left[R_t \cdot \frac{e}{f} + R_e \cdot \frac{1}{2}\left(\beta + \gamma + f^3\right) \cdot P_{d,f_{\max}} \cdot \frac{e}{f} \right]$, $f^* = \sqrt[3]{\frac{R_t}{R_e \cdot P_{d,f_{\max}}} + 1}$, then the optimal frequency of island M_1 is $f' = f_{\max}$, and the optimal cost of *PM* is $Cost_{a,\min} = 2 \cdot R_t \cdot e + 2 \cdot P_{d,f_{\max}} \cdot R_e \cdot e$.

If tasks J_1 and J_2 are executed on core $C_{1,1}$ and $C_{2,1}$ respectively, as shown in **Fig. 1 b)**, the cost of *PM* is $Cost'_a = 2 \cdot R_t \cdot \frac{e}{f} + R_e \cdot \left(f^3 + \beta + \gamma \right) \cdot P_{d,f_{\max}} \cdot \frac{e}{f}$. And the optimal frequency of islands M_1 and M_2 is f_{max}, and the optimal cost of is $Cost'_{a,\min} = 2 \cdot R_t \cdot e + 3 \cdot P_{d,f_{\max}} \cdot R_e \cdot e$. Obviously, we have $Cost_{a,\min} < Cost'_{a,\min}$, that is, fully utilization of the computation resource in an island leads to less energy.

When the set of tasks $J'=\{J_1, J_2, J_3\}$, the tasks can be scheduled as shown in **Fig. 2**. If tasks J_1 and J_2 are executed on $C_{1,1}$, and J_3 is executed on $C_{1,2}$, the cost and theoretical frequency are $Cost_b = 3 \cdot R_t \cdot \frac{e}{f_1} + R_e \cdot \left(f_1^3 + \frac{1}{2}(\beta + \gamma) \right) \cdot P_{d,f_{\max}} \cdot \frac{e}{f_1} + R_t \cdot \frac{e}{f_2} + R_e \cdot \left(f_2^3 + \frac{1}{2}(\beta + \gamma) \right) \cdot P_{d,f_{\max}} \cdot \frac{e}{f_2}$, and $f_1^* = \sqrt[3]{\frac{2 \cdot R_t}{R_e \cdot P_{d,f_{\max}}} + 1}$, $f_2^* = \sqrt[3]{\frac{R_t}{2 \cdot R_e \cdot P_{d,f_{\max}}} + 1}$. Then the optimal frequency of two islands are f_{max}, and the optimal cost is $Cost_{b,\min} = 4 \cdot R_t \cdot e + 3 \cdot P_{d,f_{\max}} \cdot R_e \cdot e$.

In schedule 2 in **Fig. 2 b)**, the cost is $Cost'_b = 3 \cdot R_t \cdot \frac{e}{f} + R_e \cdot \left(f^3 + \beta + \gamma \right) \cdot P_{d,f_{\max}} \cdot \frac{e}{f}$. Hence the optimal frequency and cost are f_{max} and $Cost'_{b,\min} = 2 \cdot R_t \cdot e + 3 \cdot P_{d,f_{\max}} \cdot R_e \cdot e$ respectively. We can see that $Cost_{b,\min} < Cost'_{b,\min}$, in other words, fully utilization of idle islands may lead to less cost.

Algorithm 1 ICAS (Voltage Island DVFS-based Cost-Aware Scheduling of Tasks)

Input: $P_{d,f_{\max}}, \beta, \gamma, m, M$, $F = \{f_1, f_2, \cdots, f_{ns}\}$ and $J = \{J_1, J_2, \cdots, J_n\}$

Output: *Schedule* and *Cost*

1 : *Schedule* = ∅ , Q_j = ∅ , *Cost* = 0; //Initialization
2 : Sort tasks with decreasing *e*; //sorting tasks
3 : For *i*=1 to min{*n*,*m*} do //task allocation for light workload
4 : *g* = (*i*-1)/*M* +1;
5 : *q* = *i* mod *M*;
6 : $Q_{g,q}$.Insert(J_i);
7 : Schedule.Insert(J_i,$C_{g,q}$);
8 : End For
9 : For *i*= *m*+1 to *n* do // task allocation for heavy workload
10 : Find the island M_j with least workload W_j;
11 : J_i is assigned to $C_{j,q}$ on island M_j with least workload $W_{j,q}$;
12 : $Q_{j,q}$.Insert(J_i);
13 : Schedule.Insert(J_i, $C_{j,q}$);
14 : End For //computing optimal frequency
15 : Sort the tasks of each active core with non-decreasing WCET; //sorting the cores
16 : Compute optimal frequency of each task using Eq.(7) and Eq.(8);
17 : Compute optimal frequency of each island; //computing optimal frequency of each group
18 : Compute *Cost*;
19 : Return *Schedule* and *Cost*

The main idea of task allocation is fully utilization of the computation resources and workload balance of the cores. The tasks are allocated as follows:

(1) If $n \le m/M$, only one voltage island is used to process the tasks, and each task is executed by an individual core in the island.

(2) If $m/M \le n \le m$, $\lceil n \cdot M/m \rceil$ voltage islands, of which at most one islands is not fully utilized, are elected to process the tasks, each task is executed by an individual core in the islands.

(3) If $n > m$, all the cores of the processor is used to process the tasks.

The cost-aware algorithm ICAS of scheduling tasks on multicore server is shown in Algorithm 1. We first initialize the variables and sort the tasks $\{J_1,J_2,\cdots,J_n\}$ according to e_i ($1 \le i \le n$) in non-decreasing order, then the tasks are allocated to the cores sequentially. Each task J_i is assigned to an idle core in a voltage island M_j until all cores of M_j are active or all tasks have allocated. This process is repeated on another idle island M_{j+1} until all tasks have completed or all the cores are active, as shown in steps 3-8. For the remaining tasks, the task will first allocated to the island M_j with the least workload and the core $C_{j,q}$ with the least increamental cost on the island, as shown in steps 9-14. Once the tasks have been allocated, we should sort the tasks on each active core with non-decreasing WCET, and the frequency of each core can be computed according to Eq.(7) and Eq.(8). Then the frequency of each island will be obtained, and finally we get the total cost of the server.

6. Performance Evaluation

We evaluate our proposed algorithm ICAS by simulated experiments. The default values of the variables in the experiments are shown in **Table 1**. The worst case execution time of the tasks are assumed to follow the normal and random distributions respectively. The workload is light if the number of tasks is less than 10, while the workload is heavy if there are more than 50 tasks. The temporal cost, energy cost and total cost of ICAS are compared to that of the algorithm CBM proposed in [14] and SFA proposed in [15].

Table 1. The default values of the variables

Variables	Meaning	Default value
m	the number of cores	8
M	the number of voltage islands	2
n	The nubmer of tasks	light workload: $1 \le n \le 10$ heavy workload: $50 \le n \le 500$
e	WCET of each task	$1 \le e \le 20$
$P_{d,max}$	Maximum dynamic power	50W
β, γ	factor of static power and other power	$\beta=\gamma=0.5$
R_t , R_e	energy factor and temporal factor	$R_t = R_e = 1$
F	The set of operating frequencies	1GHz, 1.2GHz, 1.4GHz, 1.6GHz, 1.8GHz, 2GHz, 2.2GHz, 2.4GHz, 2.6GHz, 2.8GHz, 3GHz

Fig. 3 and **Fig. 4** show the temporal cost and energy cost of ICAS and CBM for processing various number of tasks with random distribution. The results of **Fig. 3** indicate that the temporal cost and energy cost of ICAS are both less than that of CBM. The temporal cost of ICAS is linear to the number of tasks when the number of tasks is less than the number of cores since each core only processes one task. The energy cost of ICAS increases when processing

more tasks, and the energy cost of each island is linear to the number of active cores. Since only one island is used to process the tasks in CBM, the energy cost is linear to the number of tasks, while the temporal cost is proportional to the square of the number of tasks.

When the workload is heavy, all the cores are used to process the tasks in ICAS, while only one island is used in CBM. The increamental speed of temporal cost is much faster than that of energy cost until the temporal cost is a majority of the total cost. Therefore, the difference between the total costs of ICAS and CBM becomes larger for processing more tasks. Though the number of active cores in ICAS is two times more than that in CBM, the total cost of CBM is more than two times of the cost of ICAS, because the temporal cost of a queue of tasks is proportional to the square of the number of tasks in the queue.

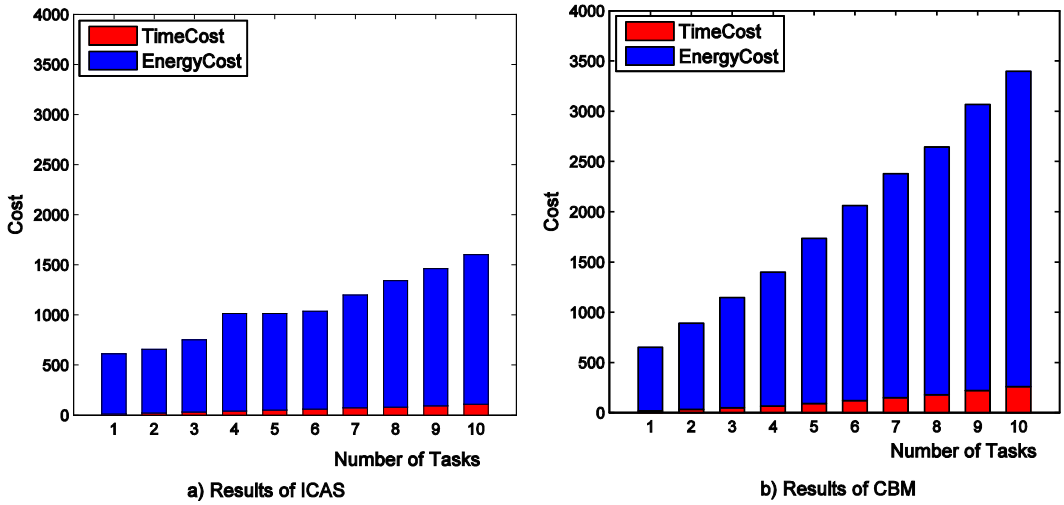


Fig. 3. Cost of ICAS and CBM for random distribution and light workload

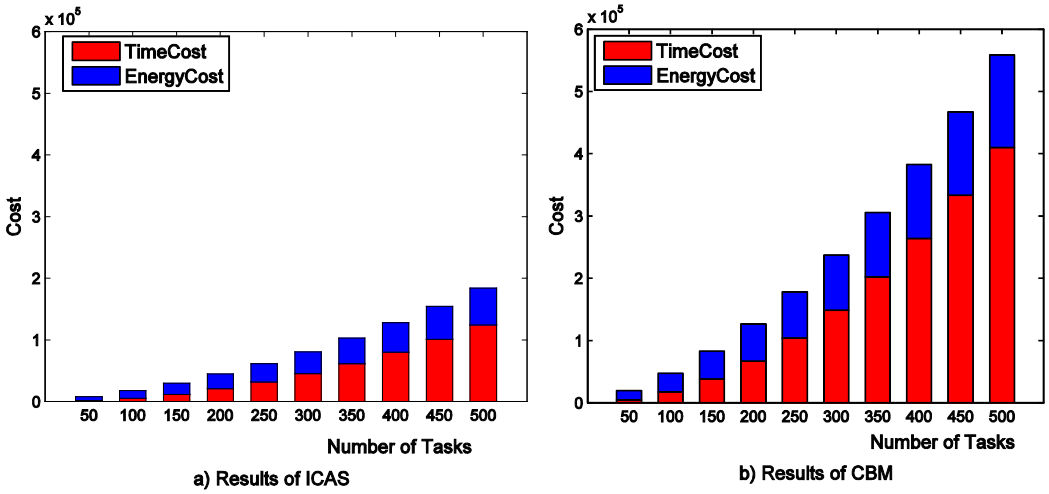


Fig. 4. Cost of ICAS and CBM for random distribution and heavy workload

Fig. 5 and **Fig. 6** show the temporal cost, energy cost and total cost of ICAS and CBM for processing tasks with normal distribution. It can be seen that the relationship between the costs and the number of tasks in normal distribution is the same as that in random distribution. The temporal cost and total cost of ICAS for processing 3, 4 and 5 tasks are (31, 40, 50) and (749, 1014, 1013) respectively, while that of CBM are (48, 64, 90) and (1144, 1397, 1734)

respectively in **Fig. 5**. The temporal cost and energy cost of ICAS for processing 100, 200 and 300 tasks are (5397, 45213, 124215) and (17571, 80879, 183724), and the percentages of temporal cost in total cost are 31%, 56% and 68% respectively, while that of CBM are (17198, 148413, 410227) and (47073, 237217, 558688) and the percentages of temporal cost in total cost are 37%, 63% and 73% respectively in **Fig. 6**.

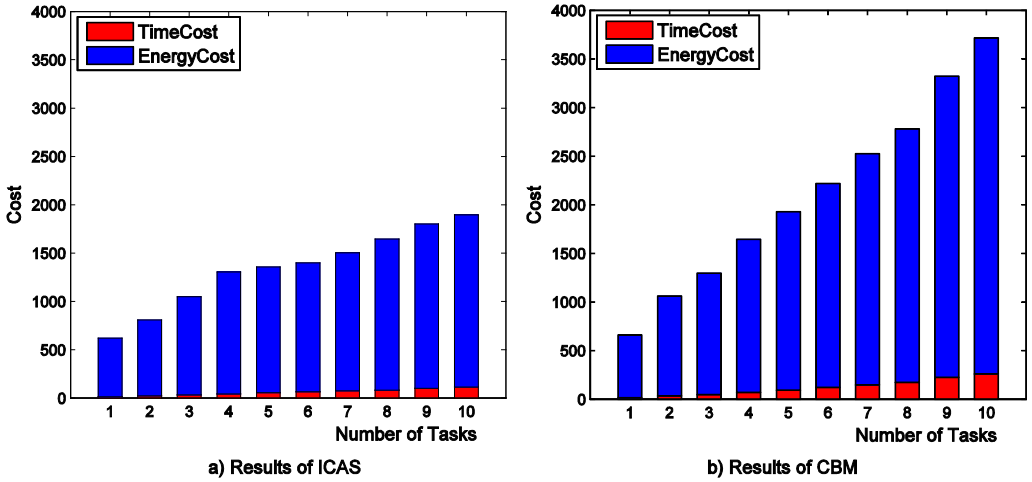


Fig. 5. Cost of ICAS and CBM for normal distribution and light workload

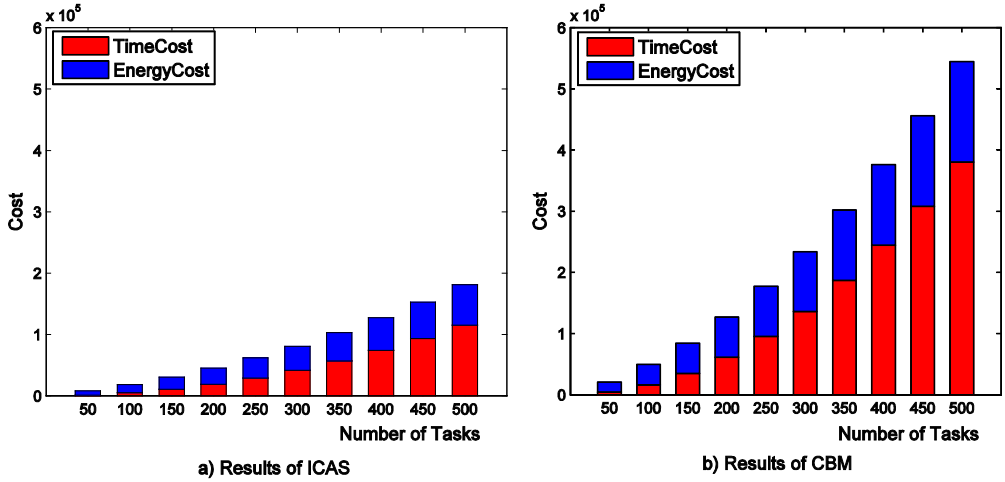


Fig. 6. Cost of ICAS and CBM for normal distribution and heavy workload

The operating frequency and the task allocation are affected by the number of voltage islands of the processor. **Fig. 7** and **Fig. 8** show the cost of the two algorithms for processing tasks in random and normal distributions respectively on the processor with 8 cores divided into 4 voltage islands. It is obvious that the temporal cost and energy cost of ICAS in the case of 4 islands are much less than that in the case of 2 islands for processing tasks with both random distribution and normal distribution. However, the temporal cost of CBM in the case of 4 islands is much larger than that in the case of 2 islands, while the energy cost remains fixed for processing tasks with both random distribution and normal distribution. It is because that all the cores are activated to process the tasks in ICAS, but only one island is active in CBM when the number of tasks is more than that of cores. Hence only two cores are activated

to process the tasks in the case of 4 islands, while 4 cores are used in the case of 2 islands in CBM. For example, the temporal costs of ICAS for processing 10, 50 and 100 tasks with random and normal distributions are (114, 1410, 5061) and (118, 1423, 5040) respectively, and the energy costs of ICAS are (1824, 6813, 13259) and (1819, 6704, 13264) respectively. As to the algorithm CBM, the temporal costs in random and normal distributions are (259, 4247, 15993) and (424, 8095, 30931), and the energy costs are (3431, 16511, 32958) and (3440, 16621, 32801) respectively.

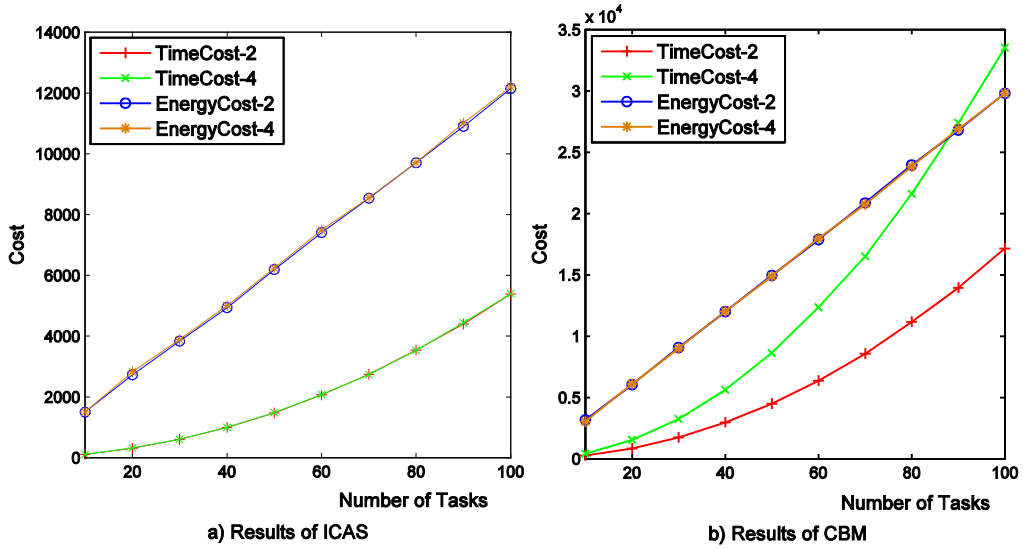


Fig. 7. Cost of ICAS and CBM for random distribution and various islands

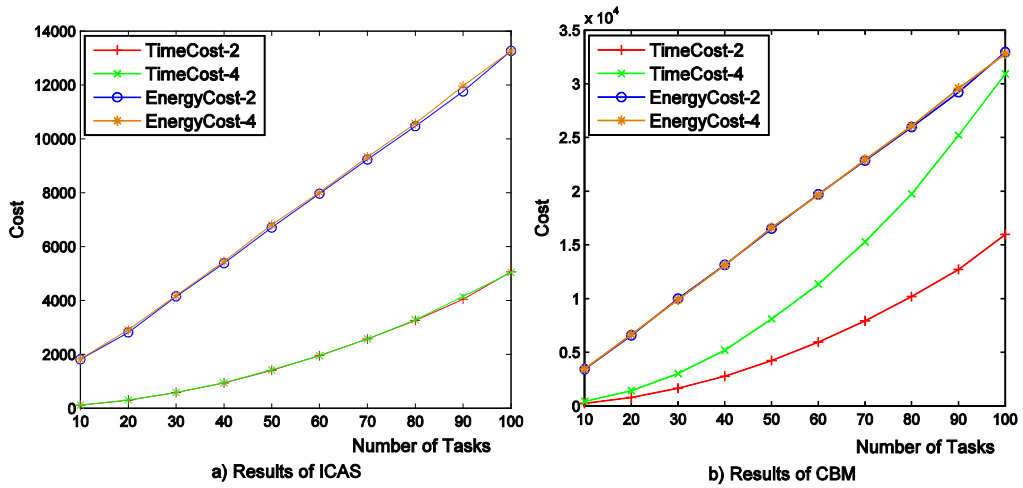


Fig. 8. Cost of ICAS and CBM for normal distribution and various islands

Fig. 9 and **Fig. 10** show the cost of ICAS and SFA for processing different tasks in random and normal distributions respectively on the processor with 8 cores divided into 4 voltage islands. The results show that the energy cost of ICAS is approximate to that of SFA, while the time cost of ICAS is higher than that of SFA. The main reason is that all cores operate on the same low frequency, and the operating frequency of the cores cannot be changed when processing tasks in SFA. Hence the waiting time of each task is longer in SFA, and the time cost of SFA is linear to the square of the number tasks according to Eq.(4). It is revealed from

the results the total cost of ICAS is less than that of SFA, and the difference will be greater with the increasing tasks regardless of the distributions.

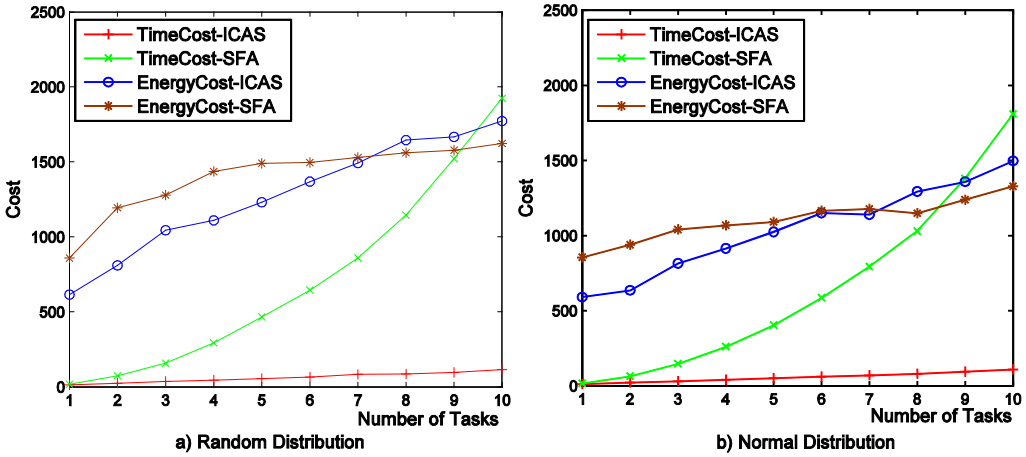


Fig. 9. Cost of ICAS and SFA for light workload

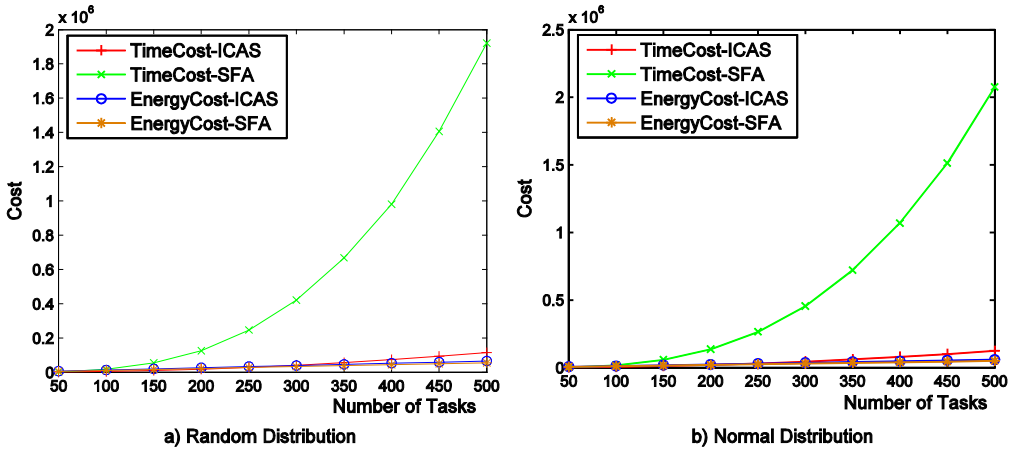


Fig. 10. Cost of ICAS and SFA for heavy workload

7. Conclusion

Multi-core server is the basic physical unit of cloud platform, and it consumes more and more energy. Since the cloud service is used in a “pay as you go” manner, the service provider must take both energy consumption and QoS into consideration to attract more users and increase the profits. A cost consisting of energy cost and temporal cost is defined, and a cost-aware scheduling algorithm ICAS is prosed to reduce the cost of processing tasks on multi-core server using voltage island-based DVFS. We assign the tasks to the cores of the server, compute the optimal frequency of each core, and find the practical frequency of each island. The experiments’ results show that then cost of ICAS is much less than existing methods.

References

[1] L. A. Barroso, "The Price of Performance," *Queue*, vol.3, no.7, pp. 48-53, 2005. Article (CrossRef Link)

[2] M. Dayarathna, Y. Wen, R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Communications Survey & Tutorials*, vol.18, no.1, pp.732-794, 2016. Article (CrossRef Link)

[3] W. Forrest. "How to cut data centre carbon emissions?," Article (CrossRef Link).

[4] L. Barroso, U. Holzle, "The Case for Energy Proportional Computing," *IEEE Computer*, vol.40, no.12, pp.33-37, 2007. Article (CrossRef Link)

[5] U.S. Energy Information Administration, "Net generation by energy source: Total (all sectors)," Article (CrossRef Link).

[6] M. Webb, "SMART 2020: enabling the lowcarbon economy in the information age, a report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI)," *Global eSustainability Initiative (GeSI) Technical report*, 2008. Article (CrossRef Link)

[7] C.C. Lin, Y.C. Syu, C.J. Chang, J.J. Wu, P. Liu, et al, "Energy-efficient Task Scheduling for Multi-core Platforms with per-core DVFS," *Journal of Parallel and Distributed Computing*, vol. 86, pp. 71-81, 2015. Article (CrossRef Link)

[8] S. Liu, Q. Qiu, Q. Wu, "Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting," in *Proc. of the conference on Design, automation and test in Europe*, pp.236-241. March 10-14, 2008. Article (CrossRef Link)

[9] S. Herbert, D. Marculescu, "Variation-Aware Dynamic Voltage/Frequency Scaling," in *Proc. of IEEE International Symposium on High Performance Computer Architecture*, pp.301-312, February 14-18, 2009. Article (CrossRef Link)

[10] C.A. Barros, L.F.Q. Silveira, C.A. Valderrama, S. Xavier-de-Souza, "Optimal processor dynamic-energy reduction for parallel workloads on heterogeneous multi-core architectures," *Microprocessors and Microsystems*, vol. 39, pp. 418-425, 2015. Article (CrossRef Link)

[11] M. Moeng, R. Melhem, "Applying Statistical Machine Learning to Multicore Voltage & Frequency Scaling," in *Proc. of the 7th ACM international conference on Computing frontiers*, pp.277-286, May 17-19, 2010. Article (CrossRef Link)

[12] H. Aydin, Q. Yang, "Energy-aware Partitioning for Multiprocessor Real-time Systems," in *Proc. of the 17th IEEE International Parallel and Distributed Processing Symposium*, pp.113.2, April 22-26, 2003. Article (CrossRef Link)

[13] R. E. Korf, E. L. Schreiber, M. D. Moffitt, "Optimal Sequential Multi-Way Number Partitioning," in *Proc. Of International Symposium on Artificial Intelligence and Mathematics*, pp.1-7, January 6-8, 2014.

[14] F. Kong, W. Yi, Q. Deng, "Energy-Efficient Scheduling of Real-Time Tasks on Cluster-Based Multicores," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition*, pp.1135-1140, March 14-18, 2011. Article (CrossRef Link)

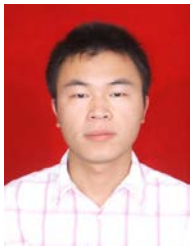
[15] S. Pagani, J.J. Chen, M. Li, "Energy Efficiency on Multi-Core Architectures with Multiple Voltage Islands," *IEEE Transactions On Parallel And Distributed Systems*, vol.26, no.6, pp.1608-1621, 2015. Article (CrossRef Link)

[16] S. Pagani, J.J. Chen, "Energy Efficiency Analysis for the Single Frequency Approximation (SFA) Scheme," *ACM Transactions on Embedded Computing Systems*, vol.13, no.5s, pp.1-25, 2014. Article (CrossRef Link)

[17] J. Liu, J. Guo, "Energy efficient scheduling of real-time tasks on multi-core processors with voltage islands," *Future Generation Computer Systems*, vol.56, pp.202-210, 2016. Article (CrossRef Link)

[18] J. Mair, K. Leung, Z. Huang, "Metrics and Task Scheduling Policies for Energy Saving in Multicore Computers," in *Proc. of 11th IEEE/ACM International Conference on Grid Computing*, pp.266-273, October 25-28, 2010. Article (CrossRef Link)

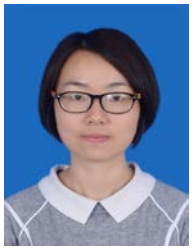
[19] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, "Power and energy containers for multicore servers," in *Proc. of the 12th ACM SIGMETRICS/ PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pp.403-404, June 11-15, 2012. Article (CrossRef Link)
[20] C. G. Tseng, S. Figueira, "An analysis of the energy efficiency of multi-threading on multi-core machines," in *Proc. of the International Conference on Green Computing*, pp.283-290, August 15-18, 2010. Article (CrossRef Link)
[21] Xinning Hui, Zhihui Du, J. Liu, Hongyang Sun, Yuxiong He and D. A. Bader, "When Good Enough Is Better: Energy-Aware Scheduling for Multicore Servers," in *Proc. of IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 984-993, May 29-June 2, 2017. Article (CrossRef Link)
[22] K. Li, "Improving Multicore Server Performance and Reducing Energy Consumption by Workload Dependent Dynamic Power Management," *IEEE Transactions on Cloud Computing*, vol.4, no.2, pp.122-137, 2016. Article (CrossRef Link)



Ding Youwei, born in 1987. He received his Ph.D degree in 2016 form Nanjing University of Aeronautics and Astronautics. Currently, he is an associated professor at the College of Information Technology, Nanjing University of Chinese Medicine, China. His research interests include energy efficient data management, cloud computing and data mining.

Liu Liang, born in 1985. He received his Ph.D degree in 2012 form Nanjing University of Aeronautics and Astronautics. Currently, he is an associated professor at at the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include wireless sensor network and data management in distributed environment.

Kongfa Hu, born in 1970. He received the M. E. degree in computer application from Anhui University of Science & Technology, Huainan, China, in 1997, and the Ph.D. degree computer application technology from Southeast University, Nanjing, China, in 2004. Currently, he is a professor at Nanjing University of Chinese Medicine and the dean of the College of Information Technology of Nanjing University of Chinese Medicine. His research interests include RFID Data Management and Data Mining, Traditional Chinese Medicine Informatics.



Caiyan Dai, born in 1985. She received her Ph.D degree in 2017 form Nanjing University of Aeronautics and Astronautics. Currently, she is an associated professor at the College of Information Technology, Nanjing University of Chinese Medicine, China. Her research interests include computer software, artificial intelligence, data mining.