International Journal of Internet, Broadcasting and Communication Vol.9 No.2 78-86 (2017)
https://doi.org/10.7236/IJIBC.2017.9.2.78

IJIBC 17-2-11

Energy-efficient Scheduling of Periodic Real-time Tasks on Heterogeneous Grid Computing Systems

Wan Yeon Lee[1*], Yun-Seok Choi[1]

[1]*Dept. of Computer Science, Dongduk Women's University, Korea*
[*]*wanlee@dongduk.ac.kr, cooling@ dongduk.ac.kr*

Abstract

In this paper, we propose an energy-efficient scheduling scheme for real-time periodic tasks on a heterogeneous Grid computing system. The Grid system consists of heterogeneous processors providing the DVFS mechanism with a finite set of discrete clock frequencies. In order to save energy consumption, the proposed scheduling scheme assigns each real-time task to a processor with the least energy increment. Also the scheme activates a part of all available processors with unused processors powered off. Evaluation shows that the proposed scheme saves up to 70% energy consumption of the previous method.

Keywords: *Real-time Task, Scheduling, Energy-efficient Design, Heterogeneous Grid*

1. Introduction

As low-cost computers and high-speed networking technologies are developed, distributed computing platforms usually comprise a heterogeneous collection of computers. A heterogeneous Grid computing system is a large loosely-coupled virtual supercomputer formed by combining many heterogeneous platforms of different characteristics. The heterogeneous Grid system is suitable for solving complex problems [1].

One of main problems when using Grid computing systems is a scheduling problem that finds an optimal schedule for a set of tasks to be executed. In the scheduling problem of real-time tasks, the goal is to assign each real-time task to the computing resources so as to finish its execution before its corresponding deadline. Another critical issue for Grid computing systems is an efficient energy and thermal management. High energy dissipation of processing components results in temperature increase of computing systems. The temperature increase directly impacts the performance and reliability of integrated circuits (ICs) [2]. Thus low energy dissipation of processing components is important for the scheduling problem of real-time tasks.

Whereas energy-efficient scheduling problems of real-time tasks on homogeneous computing systems have been widely studies [3-6], those on heterogeneous computing systems are rarely studied. A few

Manuscript Received: Apr. 17, 2017 / Revised: Apr. 28, 2017 / Accepted: May. 12, 2017
Corresponding Author: wanlee@dongduk.ac.kr
Tel: +82-2-940-4685, Fax: +82-2-940-4170
Dept. of Computer Science, Dongduk Women's University

studies [7-10] dealt with the scheduling problem of real-time tasks on heterogeneous processing components. However, they considered only the scheduling problem for *sporadic* real-time tasks, but not for *periodic* real-time tasks.

In this paper, we propose an energy-efficient scheduling scheme that minimizes the energy consumption of periodic real-time tasks while satisfying their deadlines on a Grid computing system composed of heterogeneous processor components. In the considered system, processing components provide the dynamic voltage and frequency scaling (DVFS) mechanism for efficient energy management. The DVFS mechanism dynamically changes the voltage supplied to processors. The processor speed is proportional to supplied clock frequencies, and the power consumption is approximately proportional to a polynomial function of the clock frequency [11, 12]. In practical DVFS-enabled processors, only a finite set of discrete frequencies are available and the relationship between available discrete frequencies and their power consumption is irregular.

The proposed scheduling scheme assigns each real-time task to a processor with the least energy increment. Also the scheme activates a part of all available processors with unused processors powered off after assigning all tasks to a part of processors, in order to reduce the leakage power consumption of idle processors [5, 11]. We formally solve the minimization problem of real-time tasks over finitely discrete clock frequencies with irregular power consumptions. The proposed scheme finds a near minimum-energy feasible schedule within a polynomial time, because the problem of minimizing the energy consumption of real-time tasks while meeting their deadline is NP-hard. Evaluation results show that the proposed scheme saves up to 70% energy consumption of the previous method.

The rest of this paper is organized as follows; Section 2 explains the considered system model. Section 3 describes the proposed scheme in detail. Section 4 shows evaluation results. Section 5 provides concluding remarks.

2. System Model

There are given M periodic tasks with no interdependency. A periodic task consists of consecutive instances arriving separately and sequentially. The computation of the previous instance must be completed before arrival of the next instance. The arrival period becomes the deadline of each instance. The worst computation amounts of tasks are considered for static scheduling. The instance of the m^{th} task is denoted as T_{m}. Each T_{m} requires at most C_{m} computation cycles to be executed within its deadline D_{m}. In terms of execution time, the task completion time is the required computation cycles divided by the clock frequency supplied to a processor.

The considered processors support the dynamic voltage and the frequency scaling (DVFS) mechanism that dynamically changes the clock frequency supplied to the processors. Computation speed of the processors is typically proportional to the clock frequency, and their power consumption is approximately proportional to the square of the clock frequency. In practical DVFS-enabled processors, only a finite set of discrete clock frequencies are available and the relationship between available discrete frequencies and their power consumption is irregular. For the practical DVFS evaluation, we use the data obtained from well-known DVFS processors: the Intel XScale processor and the IBM PPL405LP processor [11].

K discrete frequencies available to these processors are denoted as $f_1, f_2, \ldots, f_{\text{K}}$ in increasing order. For a given frequency f_{k} where $1 \leq k \leq K$, the power consumption is denoted as p_{k}. Then the execution time of each cycle is $1/f_{\text{k}}$ and p_{k}, respectively. If $i < j$, then $f_{\text{i}} < f_{\text{j}}$ and $p_{\text{i}} < p_{\text{j}}$. Even when a processor has no computation to

execute, the power consumption in the idle status, i.e., leakage power consumption, is strictly positive [11, 3]. The power consumption in the idle status is denoted as p_0. For convenience, we additionally define a virtual clock frequency of the idle status as f_0 and set its value to zero, because its computation speed is semantically equivalent to zero.

Table 1 shows the available frequencies and the power consumptions (energy consumption rates) of the Intel XScale processor. Table 2 shows the available frequencies and the power consumptions of the IBM PPL405LP processor.

Table 1. Intel XScale Processor

k	0	1	2	3	4	5
f_k (MHz)	0	150	400	600	800	1000
p_k (mW)	40	80	170	400	900	1600

Table 2. IBM PPC405LP Processor

k	0	1	2	3	4
f_k (MHz)	0	33	100	266	333
p_k (mW)	12	19	72	600	750

The ratio of the execution time of T_m at the maximum clock frequency to the deadline is referred to as *task utilization* and denoted as $U_m = C_m/(f_K \cdot D_m)$. Total utilization of all tasks assigned to processor P_n is referred to as *processor load* and denoted as $L_n = \sum U$. The lowest constant frequency that can execute the tasks assigned to processor P_n is referred to as *optimal frequency* and denoted as f_n^{opt}. In other words, the optimal frequency is the minimum number of computation cycles that must be executed per second in the feasible schedule. In some case, $f_n^{opt} \notin \{ f_1, f_2, \ldots, f_K \}$ because f_n^{opt} is intrinsically continuous.

The considered problem is to minimize total energy consumption of N processors while executing M periodic real-time tasks within their respective deadlines. The M independent tasks can be executed on a subset of N processors with unused processors powered off. Activated processors select their clock frequency from a finite set of discrete frequencies $\{ f_1, f_2, \ldots, f_K \}$ and change their frequency independently at any time. A schedule is called *feasible* if the M tasks are completely executed within their respective deadlines.

3. Proposed Scheduling

Figure 1 shows an example of saving energy by considering the characteristics of heterogeneous processors and by turning off power of rarely used processors. Five periodic tasks (T_1, T_2, T_3, T_4 and T_5) are given on four processors: two XScale processors (P_1 and P_2) and two PPC405LP processors (P_3 and P_4). The optimal frequencies are 400 MHz for T_1, 300 MHz for T_2, 150 MHz for T_3, 100 MHz for T_4, and 100 MHz for T_5.

Figure 1(a) shows the case where five tasks are assigned to four processors so as to minimize the maximum processor load among the four processors based on the worst-fit-decreasing heuristic of the previous method [13]. In this case, the optimal frequency of P_1 is 400 MHz, that of P_2 is 400 MHz, that of P_3 is 150 MHz, and that of P_4 is 100 MHz. Then the mean energy consumption rate of P_1 is 170 mW, that of P_2 is 170 mW, that of P_3 is 231 mW, and that of P_4 is 72 mW. In Figure 1(a), the total energy consumption rate

of the four processors is 170 + 170 + 231 + 72 = 643. Figure 1(b) shows the case where tasks are assigned by considering the different energy consumption increments of heterogeneous processors and by turning off the power of a rarely used processor. In this case, the optimal frequency of P_1 is 500 MHz, that of P_2 is 450 MHz, that of P_3 is 150 MHz, and P_4 is powered off. Then the mean energy consumption rate of P_1 is 285 mW, that of P_2 is 227.5 mW, that of P_3 is 72 mW, and that of P_4 is zero. In Figure 1(b), the total energy consumption rate of the four processors is 285 + 227.5 + 72 + 0 = 584.5. Compared with schedule of Figure 1(a), the schedule of Figure 1(b) saves the energy consumption by about 10%.

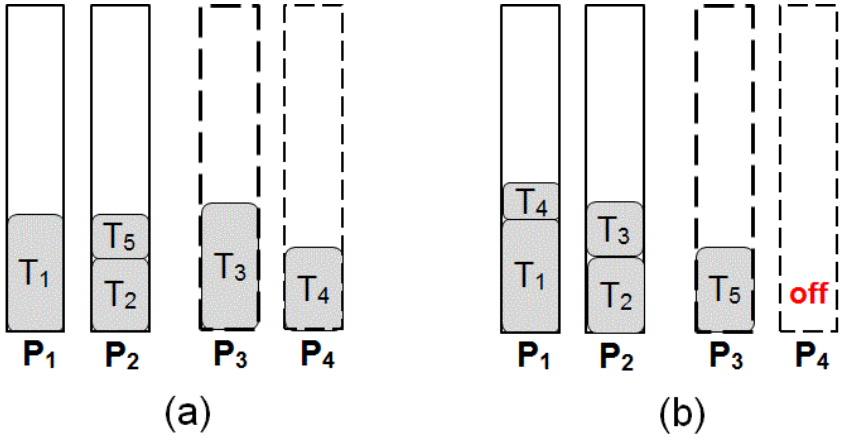

Figure 1. Working Example

Our previous study [3] verifies the Optimal Schedule of multiple periodic tasks executed on a single processor. It is known that when periodic tasks are executed on a processor, the Optimal Schedule determines their execution order according to the Earliest-Deadline First (EDF) rule and executes them at the optimal frequency f^{opt}. The optimal frequency (the lowest constant frequency executing all assigned tasks within their deadlines) of the processor P_n is derived from its processor load L_n, so as $f^{opt} = L_n \cdot f_K$.

When $L_n \cdot f_K = f_i$ for $f_i \in \{ f_1, f_2, \ldots, f_K \}$, the frequency f_i becomes the optimal frequency and the minimum-energy schedule assigns f_i to the whole computation cycles. When $f_{i-1} < L_n \cdot f_K < f_i$ for $f_{i-1} \in \{ f_1, f_2, \ldots, f_K \}$ and $f_i \in \{ f_1, f_2, \ldots, f_K \}$, the minimum-energy schedule virtually generates the optimal frequency by a combined use of the two adjacent frequencies f_{i-1} and f_i with the mean of $L_n \cdot f_K$: assigning f_{i-1} to a portion of computation cycles and assigning f_i to the rest computation cycles. The average power consumption of each activated processor P_n in the minimum-energy feasible schedule is determined by the optimal frequency $f^{opt} = L_n \cdot f_K$ and can be formulated as a function $\boldsymbol{E}_n(L_n)$ with the input L_n as follows:

$$\boldsymbol{E}_n(L_n) = \begin{cases} p_i & \text{if } L_n = \frac{f_i}{f_K} \\ p_{i-1} + \frac{p_i - p_{i-1}}{\frac{f_i}{f_K} - \frac{f_{i-1}}{f_K}} \cdot \left(L_n - \frac{f_{i-1}}{f_K}\right) & \text{if } \frac{f_{i-1}}{f_K} < L_n < \frac{f_i}{f_K} \end{cases} \quad (1)$$

The function $\boldsymbol{E}_n(L_n)$ is an increasing, convex and piece-wise linear function of L_n, such as shown in Figure 1. If $L_n > 1$, then it is impossible to complete the task before the deadline. Specially, we define $\boldsymbol{E}_n(L_n) = \infty$ when $L_n > 1$. Total power consumption of activated processors is $\sum \boldsymbol{E}_n(L_n)$.

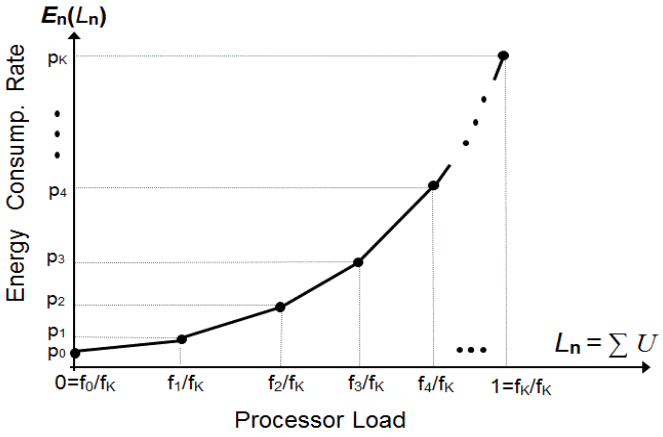

Figure 2. Relation between Processor Load and Energy Consumption Rate

Next we determine the number of activated processors among available processors. Our goal is to minimize total mean power consumption of all activated processors executing all real-time tasks. If the number of activated processors is fixed and given as η, then distributing the total utilization of all tasks ($\sum_{m=1}^{M} U_m$) evenly to η processors minimizes total power consumption of all activated processors because $\boldsymbol{E}_n(\ L_n\ )$ in Figure 2 is a convexly increasing function of L_n. The number η of activated processors is determined as follows; Define a linear function $\beta \cdot L_n$ where β is a positive constant. If β is selected to be $\beta \cdot L_n = \boldsymbol{E}_n(\ L_n\ )$ only at a unique point $L_n = \delta$ as shown in Figure 3, then $\eta = (\sum_{m=1}^{M} U_m) \ / \ \delta$. When the total utilization of all tasks ($\sum_{m=1}^{M} U_m$) is evenly distributed to η processors, the minimum power consumption of η processors is $\eta \cdot \boldsymbol{E}_n(\ \delta\ )$ because $\delta = (\sum_{m=1}^{M} U_m) \ / \ \eta$.

If $(\eta + \lambda)$ processors are activated for positive λ, then the minimum power consumption of $(\eta + \lambda)$ processors is $(\eta + \lambda) \cdot \boldsymbol{E}_n(\delta - \omega)$ where $(\delta - \omega) = (\sum_{m=1}^{M} U_m) /(\eta + \lambda)$. As shown in Figure 3, $(\eta + \lambda) \cdot \boldsymbol{E}_n(\delta - \omega) > \beta \cdot (\eta + \lambda) \cdot (\delta - \omega) = \beta \cdot \eta \cdot \delta = \eta \cdot \boldsymbol{E}_n(\ \delta\ )$. That is, the minimum power consumption of $(\eta + \lambda)$ activated processors is larger than that of η activated processors. By a similar reason, the minimum power consumption of $(\eta - \lambda)$ activated processors is larger than that of η activated processors. Hence, distributing total utilization of all tasks to η processors as evenly as possible minimizes total power consumption of all activated processors. If η is not an integer, then one of two neighboring integers is selected to have less total power consumption of activated processors.

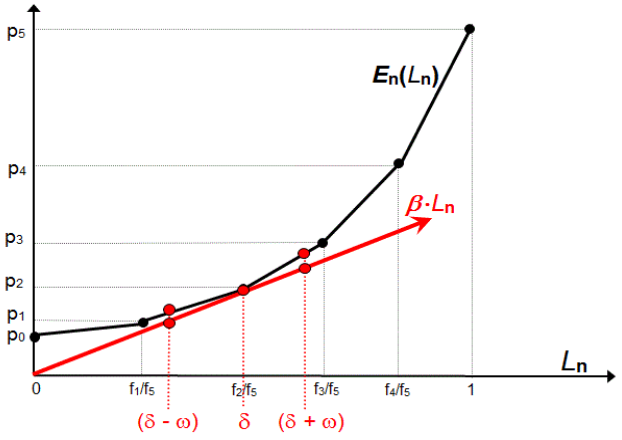

Figure 3. Working Example

When there are multiple different types of processors, we determine the number of activated processors for each type of processors. When the number of different types of processors are Φ, the numbers of activated processors for each type are denoted as $\eta_1, \ldots, \eta_\Phi$. Also the linear function shown in Figure 3 is denoted as $\beta_\alpha \cdot L_n$ and $\beta_\alpha \cdot L_n = \boldsymbol{E}_n(L_n)$ only at a unique point $L_n = \delta_\alpha$ for each α=1, …, Φ. The number η_α of activated processors for each type is calculated with distributed total utilization of all tasks. If the linear function $\beta_\alpha \cdot L_n$ has a larger value of slope, its number η_α for activated processors is calculated with a larger portion of total utilization of tasks because the processor with a larger increment of energy consumption has a larger effect of energy saving when turning off its power. Then the number η_α of activated processors for each α=1, …, Φ is calculated as follows:

$$\eta_\alpha = \beta_\alpha / (\sum_{i=1}^{\Phi} \beta_i) \times (\sum_{m=1}^{M} U_m) / \delta_\alpha \quad \text{for each } \alpha=1, \ldots, \Phi. \tag{2}$$

The remaining issue is to determine the processor to which each task is assigned. From the processor load value of each processor, we can derive the optimal frequency and the minimal long-term power consumption of each processor. The problem of minimizing total energy consumption of all processors can be formulated as follows:

$$\text{Minimize} \quad \sum_{n=1}^{N} \boldsymbol{E}_n(\boldsymbol{L_n}), \quad \text{subject to } L_n \leq 1 \quad \text{for each} \quad n \tag{3}$$

where unused processors with no assigned tasks are powered off (i.e., $\boldsymbol{E}(0) = 0$ if $L_n = 0$, instead of $\boldsymbol{E}(0) = p_0$).

Although the above problem has a lower complexity than the original task scheduling problem, it is still NP-hard for a general task set because the problem of distributing all utilization of given multiple tasks evenly to given multiple processors is NP-hard [13]. Because this computational overhead is too heavy to run even offline for a large number of available processors and tasks, we propose a scheduling scheme that finds a near minimum-energy feasible schedule within a polynomial time at the cost of a limited increment of energy consumption. The following pseudo-code describes the proposed scheduling scheme.

Step 1. Calculate task utilization values U_m for each task T_m and determine the number η_α of activated processors for each type of processors.

Step 2. Assign given M tasks to $\sum \eta_\alpha$ activated processors.

2.1: Sort all tasks in the decreasing order of their utilization values.

2.2: Assign each task one by one to the processor with the least energy increment after assignment.

Step 3. Determine the schedule of each activated processor.

3.1: Sort the execution order of the tasks assigned to each processor based on the earliest-deadline-first(EDF) rule.

3.2: Apply the optimal frequency f^{opt} (derived from the processor load L_n so as $f^{opt} = L_n \cdot f_K$) to the instant clock frequency.

Figure 4. Proposed Scheduling Scheme

The computational complexity of the proposed is $O(M \cdot log\ M \cdot N + N \cdot K)$. The complexity of Step 1 is $O(M \cdot N)$. The complexity to find the values of all β_α and δ_α is $O(M\ \ )$. The complexity to calculate the number η_α for each type of processors is $O(M\ \ )$ and that for all types is $O(M \cdot N)$. The complexity of Step 2.1 is $O(M \cdot log\ M)$ and that of Step 2.2 is $O(M \cdot N)$. The complexity of Step 3.1 is $O(M \cdot log\ M \cdot N)$ and that of Step 3.2 is $O(N \cdot K)$.

4. Evaluation

The proposed scheme is compared with the previous method [13] that assigns given tasks to all available processors without consideration of different characteristics of heterogeneous processors. It is assumed that the previous method generates the optimal frequency with a combined use of two neighboring discrete frequencies, although it was to designed to operate over infinitely continuous frequencies. As a performance metric, we define the ratio of total energy consumption in the proposed scheme to that in the previous method as *Normalized Energy Consumption* (NEC).

For performance evaluation, we employ simulation experiments with MATLAB tool on Windows 7 operating system. We use the data obtained from practical DVFS processors and synthetically generates periodic tasks. In our evaluation, it is assumed that eight Intel XScale processors and eight IBM PPL405C processors are given. Task set consists of 16, 24 or 32 periodic tasks. The deadline of each task is randomly selected between 10 milliseconds and 1 second. The number of computation cycles of each task is synthetically generated between 100,000 and 100,000,000 from a normal distribution. We run 100,000 task sets and display their average values.

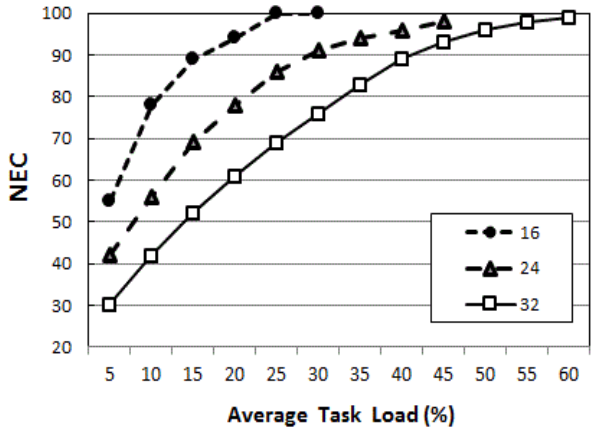

Figure 5. NEC values against average Task Load

In the first set of comparisons, we examine the performance of the relative computation amount of tasks to the deadline and the number of processors available in the system. To measure the relative computation amount of tasks to the deadline, we define the ratio of the completion time of given computation cycles under the maximum frequency of all kinds of processors to the deadline as *Task Load*, i.e., $100 \times C_m / (D_m \cdot f_K)$. Here we do not consider the time delay and the extra energy required to change the frequency at runtime.

Figure 5 shows NEC values against the average Task Load. As the average value of Task Load decreases, the energy saving effect of the proposed scheme increases. Also, as the number of given tasks increases, the energy saving effect increases. When the number of given tasks is 32 and the average Task Load is 5%, the

proposed scheme saves about 70% energy consumption of the previous method.

In the second set of comparisons, we examine the number of activated processors among all available processors in the proposed scheme. Figure 6 shows the number of activated processors among all available processors. As the average value of Task Load decreases, the proposed scheme activates fewer processors while the previous method activates all available processors. From Figure 5 and Figure 6, it is verified that the proposed scheme saves more energy as the proposed scheme activates fewer processors among given available processors.

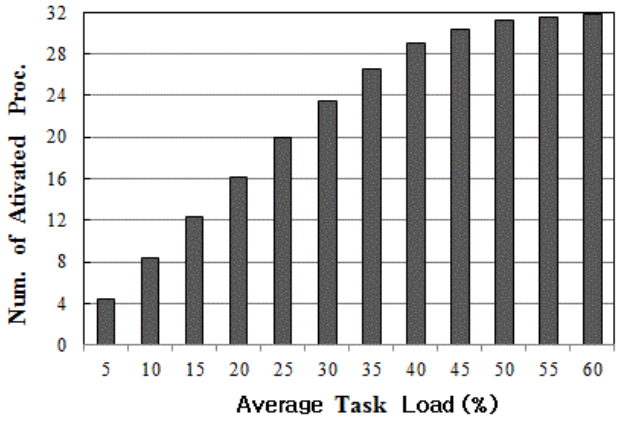

Figure 6. Number of Activated Processors

5. Conclusions

The proposed scheduling scheme tries to minimize the energy consumption of real-time periodic tasks while their deadlines on a heterogeneous Grid computing system. The grid system consists of heterogeneous processors providing the DVFS mechanism with a finite set of discrete clock frequencies. In order to save energy consumption, the proposed scheduling scheme assigns each real-time task to a processor with the least energy increment. Also the scheme activates a part of all available processors with unused processors powered off. The proposed scheme is designed to find a near minimum-energy feasible schedule within a polynomial time, because the problem of minimizing the energy consumption of real-time tasks while meeting their deadline is NP-hard. Evaluation shows that the proposed scheme saves up to 70% energy consumption of the previous method.

Acknowledgements

This research was supported by the Dongduk Women's University Grant, 2016.

References

[1] S. Nesmachnow, B. Dorronsoro, J. E. Pecero and P. Bouvry, "Energy-Aware Scheduling on Multicore Heterogeneous Grid Computing Systems," Journal of Grid Computing (Jun. 2013), vol. 11, pp. 653-680.

[2] X. Zhou, J. Yang, M. Chrobak and Y. Zhang, "Performance-Aware Thermal Management via Task Scheduling," ACM Transactions on Architecture and Code Optimization (Apr. 2010), vol. 7, no. 1, pp. 1-31.

[3] W. Y. Lee, "Energy-efficient Scheduling of Periodic Real-time Tasks on Lightly Loaded Multicore Processors," IEEE Transaction on Parallel and Distributed Systems (Mar. 2012), vol. 23, no. 3, pp. 530-537.
[4] D. Li, J. Wu, K. Li and K. Hwang, "Energy-Aware Scheduling on Multiprocessor Platforms with Devices," International Conference on Cloud and Green Computing (Sep. 2013), pp. 26-33.
[5] W. Y. Lee, "Stochastically Power-minimum Scheduling of Real-time Multicore Systems with Leakage Power Awareness," IET Electronics Letters (Jun. 2013), vol. 49, no. 13, pp. 790-793.
[6] J. Liu and J. Guo, "Energy Efficient Scheduling of Real-time Tasks on Multi-core Processors with Voltage Islands," Journal of Future Generation Computer Systems (Mar. 2016), vol. 56, pp. 202-210.
[7] K. Li, "Optimal Load Distribution for Multiple Heterogeneous Blade Services in a Cloud Computing Environment," Journal of Grid Computing (2013), vol. 11, no. 1, pp. 27-46.
[8] H. C. Liao, Y.-S. Chen and T.H. Tsai, "On-line Real-time Task Scheduling in Heterogeneous Multicore System-on-a-chip," IEEE Transaction on Parallel and Distributed Systems (Jan. 2013), vol. 24, no. 1, pp. 118-130.
[9] Y.-S. Chen and M.-Y. Chen, "On-line Energy-efficient Real-time Task Scheduling for a Heterogeneous Dual-core System-on-a-chip," Journal of Systems Architecture (Apr.-May 2013), vol. 59, no. 4-5, pp. 234-244.
[10] G. Wang, W. Li and X. Hei, "Energy-Aware Real-time Scheduling on Heterogeneous Multi-Processor," Annual Conference on Information Sciences and Systems (Mar. 2015), pp. 1-7.
[11] R. Xu, C. Xi, R. Melhem and D. Mosse, "Practical PACE for Embedded Systems," International Conference on Embedded Software (2004), pp. 54-63.
[12] W. Yuan and K. Nahrstedt, "Energy-efficient Soft Real-time CPU Scheduling for Mobile Multimedia Systems," ACM Symposium on Operating Systems Principles (2003), pp. 149-163.
[13] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," International Parallel Distributed Processing Symposium (2003), pp. 113.2.