2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing

Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Power-scalable Cluster

Aihua Liang*†‡, Limin Xiao*†, Yongnan Li*†, Zhenzhong Zhang*† Li Ruan*†
*State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
†School of Computer Science and Engineering, Beihang University, Beijing 100191, China
‡ Institute of Computer Technology, Beijing Union University, Beijing 100101, China
Email: ruanli@buaa.edu.cn, liangah@cse.buaa.edu.cn

***Abstract*—Improving the energy efficiency of high performance clusters has become important research issue. We proposed a new algorithm that reduces energy consumption of precedence constrained parallel tasks in power-scalable clusters. To reduce energy consumption without increasing the schedule length, our algorithm reclaims both static and dynamic slack time and employs different frequency adjusting techniques in different slack time. The optimal frequency is obtained through analyzing the precedence constraints of parallel tasks. We conducted experiments to compare the proposed algorithm with two other existing algorithms. Simulation results show that the proposed algorithm can get better energy efficiency without increasing the makespan.**

***Keywords*—energy aware; cluster; makespan; DVFS; DAG**

I. Introduction

As the performance of modern processor has increased, however, high energy consumption of high performance computer system has become an important and urgent problem [1]. Nowadays, performance and energy efficiency are two key criterions of modern clusters. Designing energy efficient and environmental friendly clusters is highly desirable.

Many recent high performance microprocessors are e-quipped with the Dynamic Voltage and Frequency Scaling (DVFS) technique, which allows processors to be operated at multiple frequencies under different supply voltages at run time, thereby saving energy by spreading run cycles into idle time. Parallel applications can be represented as a directed acyclic graph (DAG), called a task graph, where nodes denote the tasks with precedence constraints and the edges denote the communications between tasks. A parallel program may have some slack time due to their precedence constraints and synchronization between the tasks.

Our research is devoted to developing the scheduling algorithm which reduces energy consumption of parallel task execution by using the DVFS mechanism at slack time. We identify the slack time of tasks in different stages and scale their supply voltages to the corresponding level thus reducing the jobs energy consumption.

The rest of the paper is organized as follows: in Section II, related work has been described. Section III introduces mathematical models including a task model, and an energy consumption model. In Section IV, we present the energy-aware scheduling strategy. Simulation results are demonstrated in Section V. Finally, Section VI provides the concluding remarks and future research directions.

II. Related work

Increasing attention has been directed toward energy efficiency research for high performance clusters [2]–[6]. Among many energy saving techniques, scheduling is an efficient approach to reducing energy consumption on clusters. Nowadays, there has been a lot of research on energy efficient scheduling based on DVFS [6]–[12] or DPM(Dynamic Power Management) [13].

Slack time of a task is the interval between complete time and deadline. DVFS is a run-time power reduction technique, which has been proven to be a feasible solution to reduce processor power consumption [5]. Tasks are run at reduced voltages and clock frequencies to fill idle periods and reduce energy consumption, while providing required performance. DVFS fills the slack time by elongating computation time.

Some work applies DVFS during the communication phases of high performance computing, for example MPI [6], [7]. Kimura et al. [8] adopted DVFS at slack time of tasks. They also designed a toolkit called PowerWatch that can monitor the power and provide the control library. Ruan et al. [9] proposed an energy-efficient scheduling algorithm, named T-DVAS, for clusters. Dynamic Voltage Scaling(DVS) technique is employed to parallel tasks followed by idle processor times to conserve energy consumption without increase schedule lengths of parallel applications. Wang et al. [10] considered the Green Service Level Agreement and studied the slack time for non-critical jobs, extends their execution time using DVFS. Kim et al. [11] proposed the DVS scheduling algorithms for time-shared and space-shared resource sharing policies. Zhu et al. [12] presented a adaptive energy-efficient scheduling for aperiodic and independent real-time tasks on heterogeneous clusters with DVS, which can adjust voltage levels according to the workload variation.

DVFS exploits low frequency and voltage at the slack time. Slack time includes the static slack time of non-critical

978-0-7695-5088-6/13 $31.00 © 2013 IEEE
DOI 10.1109/HPCC.and.EUC.2013.144

tasks and dynamic slack time due to task communication and synchronization. Existing research mainly focused on the static slack time. In this paper, we not only pay attention to static slack time, but also consider the dynamic slack time. Aiming at improving energy efficiency from two respects, our scheduling method adopted different frequency scaling strategies in different stages.

III. Mathematical Model

A. Task Model

Parallel application with precedence-constrained tasks can be represented as DAG with weight. In this paper, a parallel application G is modeled as a vector (V, E, C, T) , where $V = \{v_1, v_2, ..., v_n\}$ represents a set of parallel tasks, and $E = \{e_{ij} = (v_i, v_j)|1 \leq i, j \leq n\}$ denotes a set of communication messages among parallel tasks. In all tasks, v_1 denotes the entry node and v_n is the exit node. C is the set of edge communication costs and T is the set of computation costs. The value $t_i \in T$ is defined as the required computing time of v_i. $C_{ij} \in C$ is defined as the communication cost incurred at the edge e_{ij}.

A task is non-preemptive and indivisible work unit, which may be an assignment statement, a subroutine or even an entire program. For each edge, if v_i and v_j are allocated to the same processor, the communication cost e_{ij} is set to zero. $PRED(v_i)$ is the set of immediate predecessors of v_i and $SUCC(v_i)$ is the set of immediate successors of v_i. A task allocation matrix, named U, is a $n \times m$ binary matrix denoting a mapping of n parallel tasks to m computational nodes in a cluster. Element u_{ij} in U is "1" if task v_i is assigned to processor p_j and "0", otherwise.

B. Energy Model

A cluster is represented as a set $P = \{p_1, p_2, ..., p_m\}$, where p_i denotes processor i.

Energy consumption model is represented as the sum of processors energy and interconnections energy. Let EP be the energy consumption caused by processors. The energy consumption caused by interconnections is denoted as EC. So the total energy consumption of task set can be represented as Eq. (1)

$$E = EP + EC \tag{1}$$

Let R_p be the energy consumption rate of a processor. The execution time of task i is denoted as t_i

The energy consumption rate R_p can be expressed as Eq. (2).

$$R_p = c \cdot v^2 \cdot f \tag{2}$$

Where c is the capacity of the circuit, v is the supply voltage, and f is the frequency. The capacity c is a constant parameter when processor is working. Energy consumption W of a processor is written as a product of energy consumption rate R_p and execute time of tasks. Thus, we have

$$EP = \sum_{i=1}^{n} R_p t_i = \sum_{i=1}^{n} (c \cdot v^2 \cdot f) t_i \tag{3}$$

To calculate the energy consumption of interconnects, let el_{ij} be energy consumed by the transmission of message $e_{ij} \in E$. The energy consumption of the message can be computed as a product of communicating rate R_c and the communication cost c_{ij} as Eq.(4)

$$el_{ij} = k \cdot R_c \cdot c_{ij} \tag{4}$$

k is the constant parameter. The energy consumption of a network link is a cumulative energy consumption caused by all messages delivered over the link.

The communication energy EC can be expressed as below Eq. (5):

$$EC = \sum_{i=1}^{n} \sum_{j=1}^{n} el_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} k \cdot R_c \cdot c_{ij} = k \cdot R_c \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \tag{5}$$

Because our research focuses on the DVFS technique, processor energy consumption would be reduced during the tasks execution due to processor frequency adjustment.

IV. Energy Aware Scheduling Algorithm

The key idea of DVFS is to dynamically scale the supply voltage and frequency of CPU while meeting total computation time. The parallel application has the slack time due to the precedence constraints and synchronization or communication of tasks.

The slack time can be classified into two categories [14]: Worst Slack Time (WST) and Workload-variation Slack Time (WVST). WST belongs to static slack time, which results from low processor utilization due to precedence constraints between tasks. It can be computed before task execution. WVST occurs due to execution time variation caused by data-dependent computation. It is dynamically generated. Thus, it can be known only after execution.

A novel energy aware scheduling algorithm (called NEASA) based on DVFS technique is proposed in this section. According to two kinds of slack time, our energy aware scheduling method includes two stages.

1) In the first stage, the optimum frequency of processor is calculated based on the DAG of parallel application before tasks execution. In the WST, processor frequency would be reduced to optimum frequency. The makespan of overall parallel tasks would not be affected in adjusted frequencies.
2) In the second stage, after every task finished, the execution time of the task is checked and confirms that if the real execution time is less than the original ti. then the WVST would be calculated in the run time. In WVST, the minimum allowed frequency would be adopted to reduce the energy consumption at the most extent.

TABLE I
IMPORTANT NOTATIONS AND PARAMETERS

Notation	Definition
Top-level (v_i)	the longest distance from the entry vertex to v_i
Bottom-level(v_i)	the longest distance from v_i to the exit vertex
EST(v_i)	the earliest start time of task v_i
ECT(v_i)	the earliest completion time of task v_i
LAST(v_i)	the latest allowable start time of task v_i
LACT(v_i)	the latest allowable completion time of task v_i
RCT(v_i)	the real completion time of task v_i
WST(v_i)	Worst Slack Time of task v_i
WVST(v_i)	Workload-variation Slack Time of task v_i

The objective of the scheduling is to reduce the overall energy consumption through adopting DVFS technique in both WST and WVST according to the critical path of task graph and dynamic execution variation at run time.

A. Optimum frequency setting

Parallel task scheduling should consist of grouping and allocating. Grouping is to divide the tasks of a DAG into several groups. Allocating refers to a mapping the groups on the processor. The tasks of the same group will execute on the same processor. In the grouping and allocating of proposed algorithm, the liner clustering method is employed. The key issue is the frequency setting during execution.

The important parameters are listed in Table I. The similar notations of some parameters are used by Zong in [15].

Top Level and bottom level of v_i can be calculated based on the vector (V, E, C, T) of DAG.

EST of an entry task is defined as 0. The EST of all the other tasks can be calculated in a top-down manner by recursively applying the following term on the right side of Eq.(6)

$$EST(v_i) = \begin{cases} 0 & i = 1 \\ \max\limits_{1 \le j \le n-1, e_{ji} \in E} EST(v_j) & \\ \quad + c_{ji} & 1 < i \le n \end{cases} \tag{6}$$

ECT of all tasks can be calculated as the summation of its EST and execution time from Eq. (7).

$$ECT(v_i) = EST(v_i) + t_k \tag{7}$$

Latest Allowable Completion time (LACT): the latest allowable completion time of task can be calculated in a top-down manner by applying the following term of Eq. (8).

$$LACT(v_i) = \begin{cases} ECT(v_i) & i = n \\ \min\limits_{1 \le j \le n-1, e_{ji} \in E} LACT(v_j) & \\ \quad - c_{ji} & 1 \le i < n \end{cases} \tag{8}$$

Latest Allowable Start Time (LAST): the latest allowable start time of task can be derived from LACT of task from Eq. (9).

$$LAST(v_i) = LACT(v_i) - t_i \tag{9}$$

The worst slack time of task v_i can be calculate using the following Eq. (10).

$$WST_{v_i} = LACT(v_i) - ECT(v_i) \tag{10}$$

The critical path is longest path form an entry node to an exit node. It can decide the scheduling makespan. If $LACT(v_i) = ECT(v_i)$, then vi is the critical task. All the critical tasks can form the critical path. Since the ECT and $LACT$ of critical tasks are equal, there is no slack time to scale the processor frequency. Thus, only non-critical tasks would be adopted the dynamic frequency scaling.

The shortest execution time of task i can be calculated from the following Eq. (11).

$$\min_{t_i} = ECT(t_i) - EST(t_i) \tag{11}$$

Task v_i will attain the shortest execution time if the processor of its computing node runs with the highest frequency. In contrast, the longest execution time of task v_i can be calculated from the following Eq. (12).

$$\max_{t_i} = LACT(t_i) - EST(t_i) \tag{12}$$

If the task v_i finished within the longest execution time, the makespan would not be affected. Therefore, the optimal frequency of non-critical tasks on the fly can be derived from the aforementioned parameters using the following Eq. (13).

$$f_{Optimum} = \frac{\min_{v_i}}{\max_{v_i}} f_{High} \tag{13}$$

Optimal frequency in WST can be calculated before the parallel tasks start. However, $WVST$ dynamically exists and is unknown before the tasks running. Only a task finished, the $WVST$ can be calculated from the following Eq. (14).

$$WVST_{v_i} = LACT(v_i) - RCT(v_i) \tag{14}$$

RCT is the real completion time of task v_i. If $WVST$ is great than zero, the processor frequency of computing node in $WVST$ would be scaled to lowest.

B. Energy aware scheduling algorithm

The proposed energy aware scheduling algorithm NEASA differentiates the static slack time and dynamic slack time. Firstly, the important parameters are calculated based on the vector (V, E, C, T). The optimal frequency is derived from the important parameters. At run time, once a task finished, the dynamic workload variation slack time can be calculated. In the WVST, the lowest allowable frequency would be adopted. The algorithm description is given as follows.

Algorithm 1 NEASA

Input:

Vector: (V, E, C, T)

Output:

Scheduling list, makespan, energy consumption

//Compute the optimal frequency for each non-critical task in WST
For each task $v_i \in V$
    Calculate the $EST, ECT, LAST, LACT$;
    Calculate the WST;
    Calculate the optimal frequency for v_i;
End for
//Compute the $WVST$ for each task and scale the frequency at run time
For each task v of scheduling queue
    Assigned v to P_i;
    Scale the frequency of p_i to optimal value;
End for
While (not all tasks finished)
    If Task v has finished then
        Calculate the $WVST$;
        //if there exists $WVST$
        If $(WVST > 0)$ then
            Scale the frequency to lowest value;
        End if
    End if
End while

V. Simulation Evaluation

This section presents the compressive simulation results in terms of power-performance efficiency by comparing the proposed NEASA algorithm with the existing ERA algorithm [8] and TDVAS algorithm [9] .

The metrics used for comparison are the schedule length (makespan) and the energy efficiency. The makespan is the finish time of the last task, which is the most important metric of performance that most task scheduling strategies tried to minimize it. On energy efficiency respect, energy consumption is the important metric. The energy consumption is calculated based on the energy model proposed in Section III. The parameters used are set based on Intel Pentium M. The processor number is unlimit.

A. Baseline algorithms

Now we briefly describe the two baseline algorithms: ERA and TDVAS. They all aim at saving energy by employing DVFS technique.

1) ERA [8]. ERA is applied to parallel programs. The worst slack time is reclaimed by changing the voltage and frequency. The algorithm aims to not increase the overall execution time and allow the tasks to be execute as uniformly as possible in frequecny. The workload-variation slack time is not considered in this algorithm.
2) TDVAS [9]. TDVAS is proposed to leverage processor idle time to lower processor voltages. It only focus on worst slack time. The optimal frequency of processor is derived by the parameters based on DAG. The workload-variation slack time would not be considered.

TABLE II
Applications used to drive simulation

Application	Number of tasks
Random	100
Sparse matrix solver	96
SPEC fpppp	334

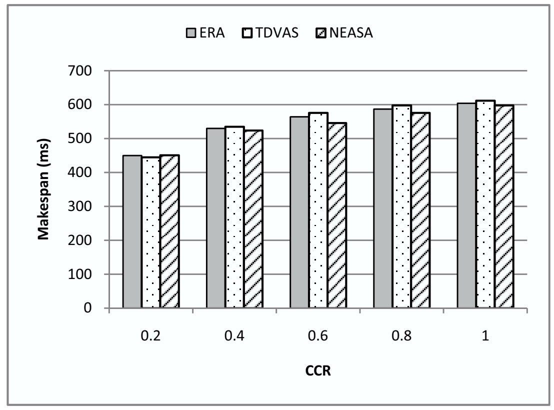

Fig. 1. Makespan comparison of random in different CCR

B. Overall energy efficiency

Communication-Computation-Ratio (CCR) is an important parameter to represent the characteristic of a parallel program. CCR denotes the ratio of communication time and computation time. Computation-intensive application has small CCR value. In contrast, communication-intensive application has large CCR value. The proposed algorithm focuses on reducing the computation energy. Therefore, CCR is varying in a reasonable range of 0 to 1 in the simulation.

Let us compare the overall performance energy efficiency of the proposed NEASA algorithm against ERA and TDVAS. The standard task graph set is a kind of benchmark for evaluation of multiprocessor scheduling algorithms [16]. We use three applications with different types. The applications used to drive simulation are listed in Table II.

We observe from Fig. 1 and Fig. 2 that for random application NEASA has the best energy efficiency among three algorithms. It can improve the energy efficiency by about 4 percent. As for makespan, three algorithms have close values.

Fig. 3 and Fig. 4 show the makespan and energy consumption comparison of sparse matrix solver, which is a communication intensive application. We can see that the differences of both makespan and energy consumption are very small. Since the proposed algorithm aims at reducing the computation energy, the reduced energy is not obvious.

Fig. 5 and Fig. 6 show the makespan and energy consumption comparison of SPEC fpppp application. We can observe that the proposed algorithm NEASA can obviously better

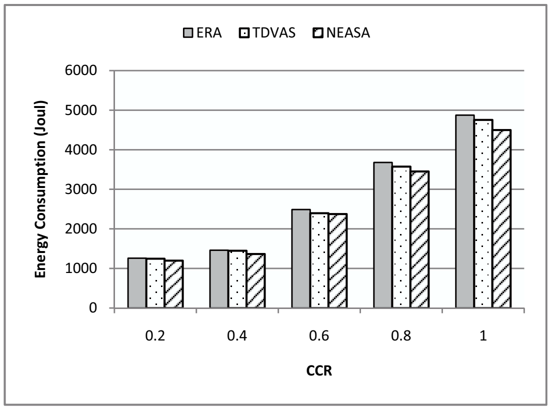

Fig. 2. Energy consumption comparison of random in different CCR

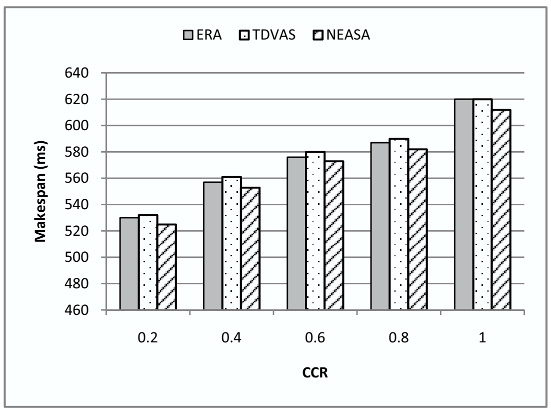

Fig. 3. Makespan comparison of sparse matrix in different CCR

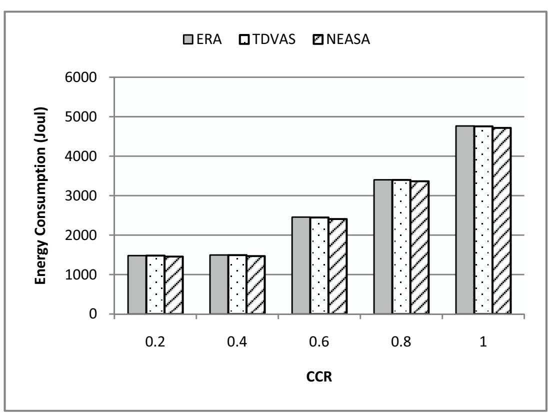

Fig. 4. Energy consumption comparison of sparse matrix in different CCR

than other two algorithms with the very close makespan. The improved energy saving can reach about 10 percent.

The proposed algorithm considers the energy reducing in the workload variation slack time. Therefore, if the parallel application is computation-intensive and has more dynamically generated slack time. The proposed algorithm would have

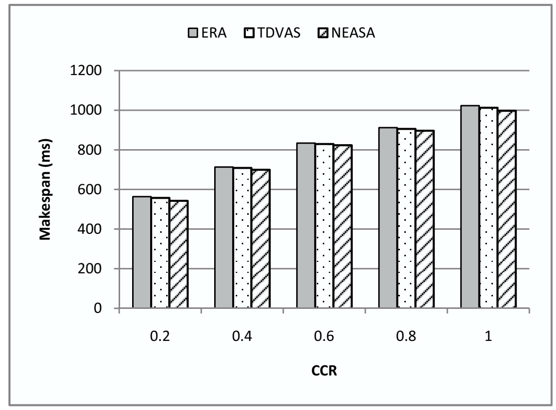

Fig. 5. Makespan comparison of fpppp in different CCR

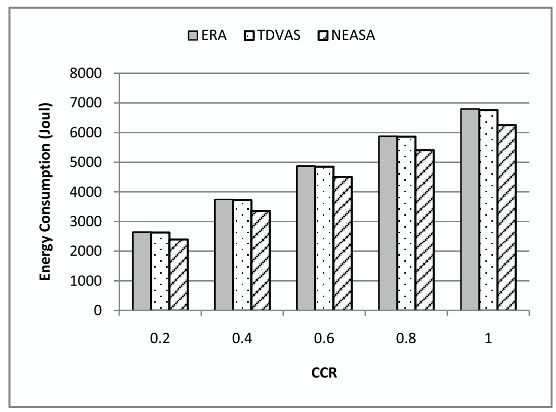

Fig. 6. Energy consumption comparison of fpppp in different CCR

better energy efficiency.

VI. Conclusion and Future Work

In this paper, we have proposed an energy-aware dynamic clustering scheduling strategy based on DVFS for parallel tasks running on clusters with an objective of reducing computing energy. For maximum energy conservation, proposed algorithm differentiates different slack time based on static and dynamic characteristics of tasks and employs different frequency scaling strategies. Optimal frequency is derived from the important parameters of DAG. Experimental results show that the proposed algorithm NEASA can effectively reduce the energy consumption compared with existing two scheduling algorithms without increasing the schedule length. It can maximally conserve overall energy consumption by 10 percent.

Future research will investigate the scheduling method to reduce the communication energy on heterogeneous clusters where computational nodes have different processing capabilities and network interconnection may have various performances.

ACKNOWLEDGMENT

This study is supported by the Hi-tech Research and Development Program of China (863 Program) under Grant No. 2011AA01A205, the National Natural Science Foundation of China under Grant No. 61232009, 61003015 and 61370059, the Doctoral Fund of Ministry of Education of China under Grant No. 20101102110018, the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2012ZX-23 and Beijing Natural Science Foundation under Grant No. 4122042. Corresponding author is Li Ruan.

REFERENCES

[1] Rong Ge, Xizhou Feng, and Kirk W. Cameron, “Performance-constrained Distributed DVS Scheduling for Scientic Applications on Power-aware Clusters”. In Proceedings of International Conference for High Performance Computing, Networking and Storage (SC’05), 2005.

[2] G. Laszewski, L. Wang, A.J. Younge, X. He, “Power-aware scheduling of virtual machines in DVFS-enabled clusters”. In Proceedings of IEEE Intl Conf. Cluster Computing, August 2009, pp. 1C10.

[3] V. Nlis, J. Goossens, R. Devillers, D. Milojevic, N. Navet, “Power-aware real-time scheduling upon identical multiprocessor platforms”. In Proceedings of 2008 IEEE Intl Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing, SUTC 2008, June 2008, pp. 209C216.

[4] L. Hu, H. Jin, X. Liao, X. Xiong, H. Liu, “Magnet: a novel scheduling policy for power reduction in cluster with virtual machines”. In Proceedings of 2008 IEEE Intl Conf. Cluster Computing, CLUSTER 2008, September 2008, pp. 13C22.

[5] C. Hsu and W. Feng, “A power-aware run-time system for highperformance computing”. In Proceedings of the 2005 ACM/IEEE conference on Supercomputing. IEEE Computer Society Washington, DC, USA, 2005.

[6] V. Freeh and D. Lowenthal, “Using multiple energy gears in MPIprograms on a power-scalable cluster”. In Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming. ACM New York, NY, USA, 2005, pp. 164C173.

[7] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, “Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs”. In Proceedings of the 2006 ACM/IEEE conference on Supercomputing. New York, NY, USA, 2006.

[8] Kimura H, Sato M, Hotta Y, Boku T, Takahashi D, “Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster”. In Proceedings of IEEE International Conference on Cluster Computing, 2006, pp. 1-10.

[9] Xiaojun Ruan, Xiao Qin, Ziliang Zong, Kiranmai Bellam, Mais Nijim, “An Energy-Efficient Scheduling Algorithm Using Dynamic Voltage Scaling for Parallel Applications on Clusters”. In Proceedings of the 16th IEEE International Conference on Computer Communications and Networks, Honolulu, Hawaii, A ug. 2007.

[10] Wang L, Von Laszewski G, Dayal J, Wang F, “Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS”. In Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, pp. 368-377.

[11] Kim K H, Buyya R, Kim J, “Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters”. In Proceedings of the seventh IEEE international symposium on cluster computing and the grid, 2007, pp. 541-548.

[12] Zhu X, He C, Li K, Qin X, “Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters”. Journal of Parallel and Distributed Computing. vol. 72, pp. 751-763, 2012.

[13] Aihua Liang, Limin Xiao, Li Ruan, “Adaptive workload driven dynamic power management for high performance computing clusters”. Computers & Electrical Engineering. http://dx.doi.org/10.1016/j.compeleceng.2013.04.026.

[14] Sanjeev Baskiyar , Rabab Abdel-Kader, “Energy aware DAG scheduling on heterogeneous systems”. Cluser computing, vol. 13, pp. 373-383, 2010.

[15] Ziliang Zong, Manzanares A, Xiaojun Ruan, Xiao Qin, “EAD and PEBD: Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters”. IEEE Transactions on Computers, vol. 60, no. 3, pp. 360-374, 2011.

[16] Standard Task Graph Set. http://www.kasahara.elec.waseda.ac.jp/ schedule/