CSIT (June 2013) 1(2):175–181
DOI 10.1007/s40012-013-0012-6

ORIGINAL RESEARCH

An intelligent energy optimization approach for MPI based applications in HPC systems

Bhavyasree Unni · Nazia Parveen · Ankit Kumar · B. S. Bindhumadhava

Received: 4 November 2012 / Accepted: 26 March 2013 / Published online: 16 April 2013

Abstract Energy-aware computing is gaining more and more attention in high performance computing (HPC) environment. As an outcome of this, various energy-aware techniques are existing and many are being proposed. But it is difficult to have a technique which saves energy without compromising the performance. This paper talks about a novel energy optimization approach for Message Passing Interface (MPI) applications running on HPC systems. Our approach relies on applying Dynamic Voltage Frequency Scaling (DVFS) at node level by an optimization agent. Whenever MPI processes are idle or busy with I/O operations, the corresponding CPU cores run at higher frequencies, which results in wastage of power. During this time, CPU cores frequencies can be reduced using DVFS so that the energy can be saved. Our approach is based on a Multi-agent based intelligent energy management framework, which uses an optimization agent for implementing energy optimization algorithm. The key advantage of the proposed approach is that the performance will not be compromised while achieving energy savings.

Keywords HPC · Energy-aware computing · MPI · DVFS · Multi-agent system · Autonomic computing

B. Unni (✉) · N. Parveen · A. Kumar · B. S. Bindhumadhava
Real Time Systems and Smart Grid Group, Centre for Development of Advanced Computing, C-DAC Knowledge park, Bangalore, India
e-mail: bhavyasreeu@cdac.in

N. Parveen
e-mail: naziap@cdac.in

A. Kumar
e-mail: ankitk@cdac.in

B. S. Bindhumadhava
e-mail: bindhu@cdac.in

1 Introduction

Enhancing the performance was the key concern in the area of high performance computing (HPC) during past years, whereas energy management was in second place. But now the scenario has reversed and the energy management has emerged as the most considerable aspect in HPC world. HPC systems consume power in several megawatts [1] and this high power consumption may lead to problems like reduced reliability, increased cost, less stability etc. Hence reducing power consumption for high end computing becomes a crucial issue at both system level and application level.

Generally, HPC systems are of distributed memory architecture and MPI standard [2] is one of the most commonly used parallel programming paradigm in this type of memory architecture. Our approach is based on applying DVFS at node level by our energy optimization tool, which depends on following two conditions. First one is, whenever the master process is executing and the slave processes are in waiting state. Second one is, when all processes are carrying out the I/O operations. In both the conditions, the corresponding CPU cores run at higher frequencies that simply wastes the CPU cycles which in turn, results in inefficient use of power. Hence, if we can reduce the power consumption during this time, this power wastage can be minimized. Our approach uses this principle to minimise the power wastage.

Dynamic power consumption of processor is proportional to the product of square of voltage and frequency [3]. During the idle period, the dynamic power consumption can be reduced by using DVFS technique. The major issues, when the frequency of processor is to be varied and on which nodes are being addressed by our energy

optimization algorithm. We have evaluated our optimization algorithm using Intel MPI Benchmarks (IMB) [4] and a pseudo code which has been developed by us.

The proposed Multi-agent based autonomic framework is composed of autonomic components (agents) interacting with each other. An autonomic computing system [5] makes decisions on its own and constantly checks and optimizes its status automatically, adapting itself to the changing conditions. Our optimization agent is self-optimizing, i.e. it will monitor the system continuously and optimizes depending on the system status automatically. This framework provides intelligence to our optimization approach so that human intervention can be avoided.

In this paper, we have used both technologies so that energy is intelligently managed using DVFS by optimization agent that will take the decision when and where to apply DVFS on nodes.

The rest of the paper is organized as follows. Section 2 reviews the related works. In Sect. 3, we explained the energy optimization problem and the algorithm to solve it. We have presented the intelligent energy management framework for energy optimization in Sect. 4. In Sect. 5, experimental analysis is discussed. Finally, Sect. 6 describes the conclusions and future works.

2 Related works

Recently, there are lots of researches being carried out in the field of power optimization in HPC systems. In this section, we focus on power optimization in the area of MPI applications

Most of the node level energy management techniques are based on DVFS techniques [6–8], because CPU is the most power consuming component within a node [9]. In [10], DVFS technique is applied to the nodes with less computation so as to reduce the power. There are some research works which are based on the energy efficient task allocation of MPI jobs. Y. Ma et al. [11] explains how efficiently task clustering with task duplication can be done to reduce energy consumption. The paper [12] discusses about task aggregation to save energy. Several researches are being carried out in reducing the CPU frequency during the communication phase of MPI programs. Dong et al. [13] focused on scaling down the CPU frequency during the MPI collective operations. Chen et al. [14] presented an Automatic Energy Status Controlling which can control CPU frequency automatically based on the communication latency in the nodes. Our paper is also concerned about applying DVFS techniques for energy optimization in MPI applications, but it mainly focuses on the state of the processes under execution.

3 Problem statement

MPI is a common parallel programming interface which distributes the task among multiple processors. Processors execute these tasks and communicate with each other by message passing. MPI is based on distributed memory model where every process has its own memory space which cannot be accessed by other processes. Basically, two types of MPI programming models are available, i.e. SPMD (Single Program Multiple Data) and MPMD (Multiple Program Multiple Data) or *Master- Slave*.

Our technique is applicable in two situations. First one is based on MPMD model, in which, usually when the master process is executing its task, all the slave processes are in waiting state. During this execution time, all of the slave processes are idle and are wasting the CPU cycles. Even though they are in waiting state, processors operate at high frequencies. Since processors run at high frequencies, this leads to higher power consumption. Our idea is to reduce the processors frequencies on slave nodes as long as slave processes are idle, so that the power wastage can be minimized.

Second situation is when processes carry out the I/O operation, all the nodes on which these processes run, operate at higher frequencies. The percentage of CPU utilization is low during execution. Hence, if we can reduce the power consumption during this time, power wastage can be minimized.

3.1 Energy model

All the modern processors are enabled with DVFS technique. This section describes DVFS enabled system model in terms of energy consumption.

DVFS enabled processor can work on sets of different voltage and frequency as given in (1) and (2).

$$V = v_{i,} \text{ wheremin} < i < max \tag{1}$$

$$F = f_{i,} \text{where } \min < i < \max \tag{2}$$

v_i is the ith operating voltage. f_i is the ith operating frequency.

The energy consumption of a processor is the sum of static energy and dynamic energy consumption and is given in (3). Energy consumption in terms of static and dynamic power is shown in (4).

$$E = E_{dynamic} + E_{static} \tag{3}$$

$$E = \left(P_{dynamic} + P_{static}\right).\Delta t \tag{4}$$

According to [15], total energy consumption equation can be modified as (5).

$$E = \left(A\ C\ v^2 f + v\ I_{leak}\right).\Delta t \tag{5}$$

where, A is the percentage of active gates, C is the capacitance load of all gates, v is the operating voltage, f is the processor frequency, I_{leak} is the leakage current, Δt is the time duration.

Unlike dynamic power, static power is not activity based. By reducing the processors frequencies when they are in idle state, the static power consumption cannot be decreased. On the other hand, shutting off the inactive part of the system does help, but it results in loss of state. DVFS may be used to reduce the dynamic power consumption by changing the CPU clock frequency-voltage setting without affecting the execution time.

3.2 Energy optimization algorithm

In this section, we introduce a novel energy optimization algorithm for MPI applications in HPC environment. This algorithm makes use of two factors, i.e. the difference in the execution time of master and slave processes and the time taken to complete the I/O task. The amount of energy that can be saved depends on the type of application. Usually, in HPC environment nodes are generally not shared among different applications that is whole node is utilized by a single job.

The workflow of this algorithm is depicted in Fig. 1. First step of the algorithm is to identify the nodes which has been allocated for a particular application with the help of scheduler. This information can be taken from the scheduler. In the next step, the algorithm will verify whether all the processes are carrying out I/O operations. If it is true, then frequency of all the nodes will be reduced. Otherwise, it will check for the second condition, i.e. whether master is executing with slaves on waiting. If this condition is satisfied, then the node on which the master is running is found by interacting with the application. Next, the frequencies of the slave processors are reduced when they are in waiting state by using DVFS. Whenever slaves start running or the I/O operations are over, the frequencies of the processors will be increased by DVFS in the last step. This algorithm will continue with the above steps and will be terminated when the application finishes.

4 Intelligent energy management framework for energy optimization

We have designed and implemented an Intelligent Energy Management framework which is based on Multi-agent support as shown in Fig. 2. A Multi-agent framework [16] consists of loosely-coupled computational autonomous agents that can perform actions. These have resources at their disposal and they possess knowledge. They are situated in a common environment and they can communicate

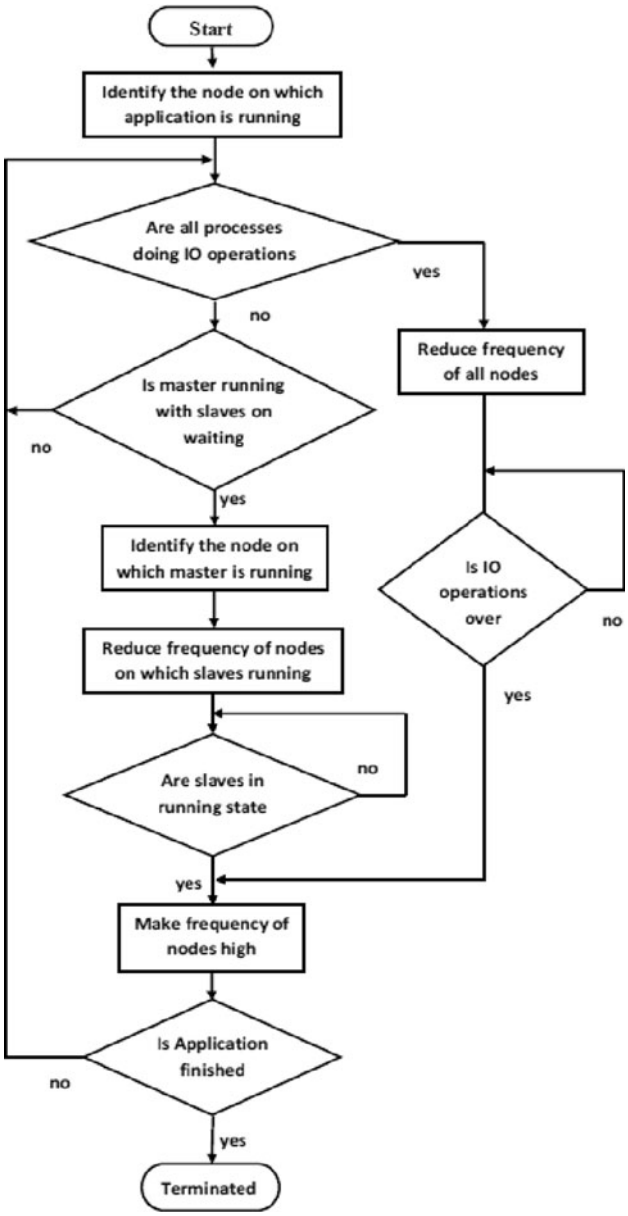

Fig. 1 Flowchart of energy optimization approach

through interaction protocols. We have used C-DAC Multi-agent Framework (CMAF) [17] to provide the support for agent execution in our architecture.

For this architecture, we have used a hybrid (reactive and mobile) type agent. A reactive agent receives input, processes it and produces output. A mobile agent is a complete self-contained body of code, which physically moves from one computer to another. Before migrating, the mobile agent stops execution at the source and resumes execution after reaching the destination.

This framework mainly consists of the Target System (TS), which is HPC System's compute nodes and Intelligent Energy Manager (IEM). The framework is deployed on HPC system where IEM is deployed at Head Node and

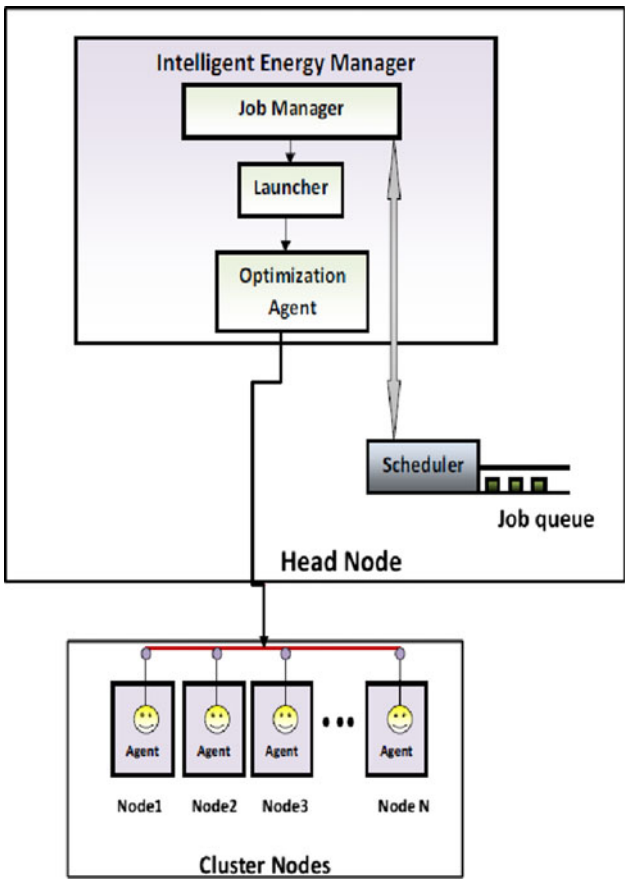

Fig. 2 Architecture of intelligent energy management framework

each compute node (which works as TS) hosts an optimization agent that executes the energy optimization algorithm. Our IEM comprises of three parts, i.e. Job Manager, Launcher and Optimization agent. Job Manager interacts with the scheduler to get details about the jobs, i.e. on which nodes each job has been allocated. It collects and updates the information regarding each job from the scheduler periodically. For every job, it passes the corresponding information to the Launcher. Based on this information, Launcher initiates the optimization agents on appropriate nodes. It also passes the corresponding parameters to each agent. At node level, this agent interacts with the application and carries out the optimization according to the algorithm mentioned in Sect. 3.2. The optimization agent is terminated with the end of application.

5 Experimental analysis

In this section, we evaluated the energy savings obtained with our energy optimization algorithm. The performance of energy optimization algorithm varies according to the nature of application i.e. whether it is CPU bound or I/O bound, duration of execution, no. of processes and nodes

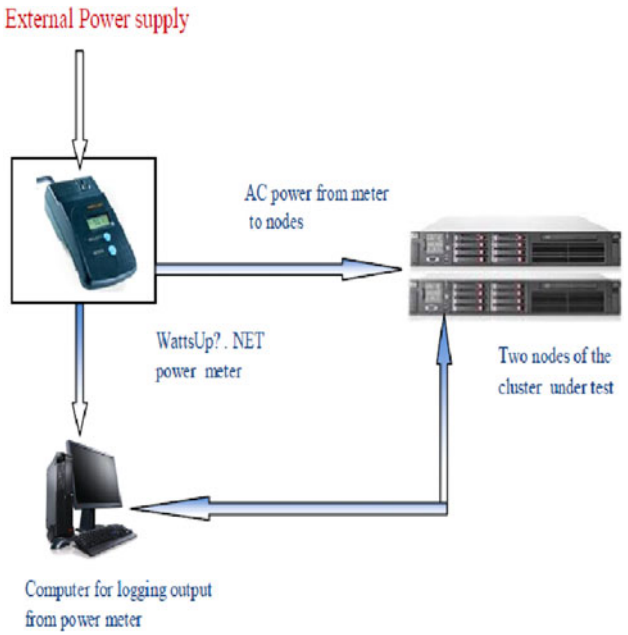

Fig. 3 Experimental setup for power measurement

etc. We have carried out our experiments with MPI-I/O benchmarks of IMB package and our pseudo code. The following subsections describe the details of the experimentation done.

5.1 Experimental environment

Our experimental platform is equipped with three HP DL380G7 servers, each having two Intel Xeon E5645 processors with six cores. These three systems are clustered using PBS resource manager and Maui scheduler where one acts as a head node and the other two as compute nodes. Each CPU core has maximum frequency of 2.4 GHz and minimum frequency of 1.6 GHz. Each node has RHEL 6.2 operating system and uses MPICH2-1.4.1 library for MPI [18]. Our Multi-agent framework is loaded into the head node and compute nodes.

Power measurement has been done using *"WattsUp?.NET"* power meter. The energy consumption is estimated by integrating the actual power measures over time. The experimental set up for power measurement is shown in Fig. 3.

5.2 Experimental results

We have evaluated the energy optimization algorithm using two experiments in high performance mode. We carried out first experiment with the pseudo code, which is based on MPMD model of MPI. It is CPU intensive and does matrix multiplication.

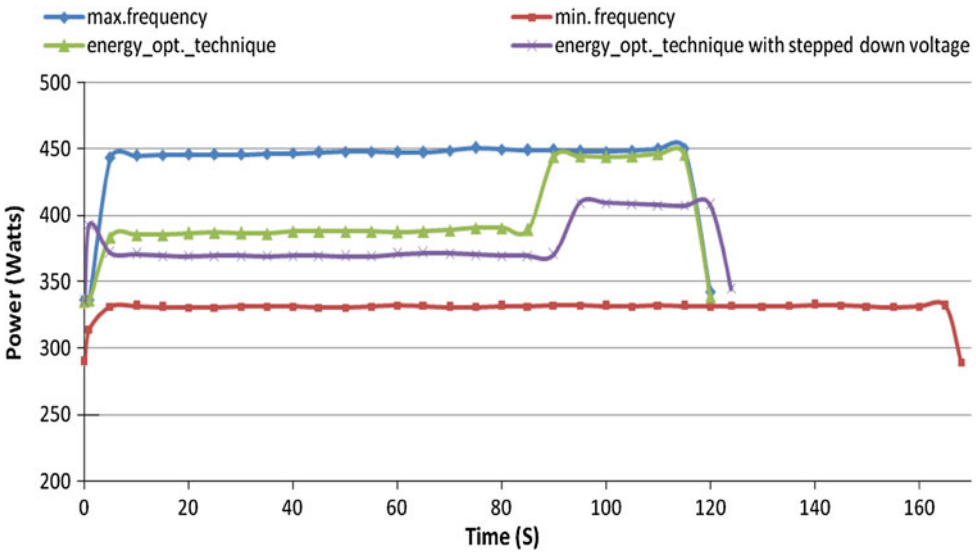

Fig. 4 Power consumption with pseudo code

Table 1 Energy consumption and savings under different conditions

	With maximum frequency	With minimum frequency	With energy optimization technique	With energy optimization technique and stepped down voltage
Avg. Power (W)	443.33	330.66	402.11	378.96
Energy (Ws)	53,643.3	55,881.5	48,656	47,370.6
Energy savings	0 %	−4 %	9.3 %	11.7 %

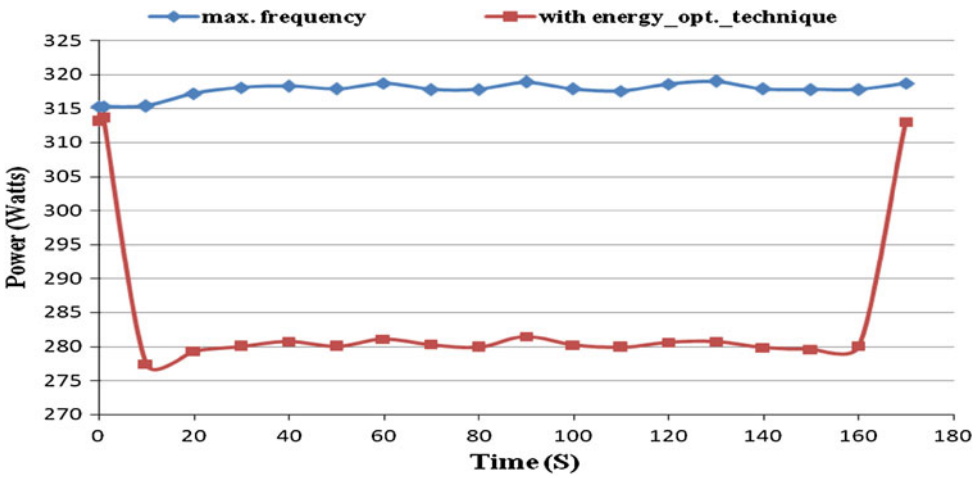

Fig. 5 Power consumption with P_write_priv under different condition

We have executed this program on two servers with 24 processes by running 12 processes on each server. We have executed the program under four different conditions, i.e. all processors operating at max. frequency (2.4 GHz), all processors operating at min. frequency (1.6 GHz), all processors at max. frequency and with our energy optimization technique, and all processors at max. frequency and with our energy optimization technique combined with varying voltage levels. We stepped down the voltage level of processors at maximum frequency by one step. In our energy optimization technique, whenever the master process is executing and the slave processes are in waiting state, the frequency of the nodes, on which slaves are running, is reduced from 2.4 to 1.6 GHz. The frequency is increased back to 2.4 GHz when the slave processes start to execute. The power consumption during all the conditions for this experiment is shown in Fig. 4.

The energy consumption and savings for the above experiments are shown in the Table 1. By utilizing our energy optimization technique, we are able to reduce the

Table 2 Energy consumption and savings under different conditions

	With max. frequency	With our energy optimization technique
Avg. Power (W)	317.8	280.5
Energy (Ws)	54,355	47,970.7
Energy savings	0 %	11.7 %

energy consumption by 9.3 % without affecting the performance. We achieved 11.7 % energy savings with 3 % increase in execution time by combining our optimization technique with varying voltage levels of processors.

The second experiment was conducted with P_write_priv benchmark. It is one of the I/O benchmark of IMB package. In this case, all participating processes perform concurrent I/O to different, private files. We have executed this program on two servers with 24 no. of processes, 12 on each server. This experiment was carried out with two different conditions, i.e. all processors with max. operating frequency (2.4 GHz) and with our energy optimization technique.

Whenever all processes are performing the I/O operations, the processor frequencies of all the nodes will be reduced from 2.4 GHz to 1.6 GHz in our energy optimization technique. The experiment results with the two conditions during the execution of P_write_priv benchmark is shown in Fig. 5. The energy consumption for both experiments is calculated by integrating power readings over the execution time. The corresponding energy consumption and savings for this experiment are shown in Table 2. By utilizing our energy optimization technique, we could reduce the energy consumption by 11.7 % without affecting the performance.

As the number of processes and the number of nodes increase, more energy can be saved.

6 Conclusions and future work

In HPC environment, enormous amount of energy wastage occurs at the application level. Therefore, the need for an efficient energy optimization algorithm is increasing tremendously. Most of the techniques are based on DVFS, which is a proven effective way to reduce power wastage. Our research is based on minimizing the energy wastage using DVFS in MPI applications. The proposed energy optimization policy is effective and can automatically set the frequency of processors, which in turn leads to reduction in energy consumption without degrading the performance. We have also developed a Multi-agent based autonomic framework which helps to implement our algorithm on HPC systems.

In future, we will deploy this algorithm using Multi-agent framework on live HPC systems running MPI applications.

Acknowledgments The authors would like to thank R. K. Senthil Kumar, H. V. Raghu, Sumit Kumar Saurav, Manisha Chauhan and B. Jayanth for their valuable support and suggestions while conducting this research.

References

1. Ge R, Feng X, Pyla H, Cameron K, Feng KW (2007) Power measurement tutorial for the Green500 List June 27, 2007
2. The Message Passing Interface (MPI) Standard. http://www-unix.mcs.anl.gov/mpi/
3. Ge R, Feng X, Cameron KW (2005) Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In: Proceedings of ACM/IEEE conference on supercomputing (SC '05)", 2005
4. Intel MPI Benchmarks- Users Guide and Methodology.http://www.software.intel.com/en-us/articles/intel-mpi-benchmarks/
5. Tewari V, Milenkovie M (2006) Standards for autonomic computing. In: Intel Technology Journal, 2006
6. Wang L, Laszewski G, Dayal J, Wang F (2010) Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In: Proceedings of IEEE symposium on cluster, cloud and grid Computing, CCGrid, 2010
7. Li D, Supinski BR, Schulz M, Cameron K, Nikolopoulos DS (2010) Hybrid MPI/OpenMP power- aware computing. In: Proceedings of IEEE symposium on parallel and distributed processing, IPDPS 2010
8. Cameron KW, Ge R, Feng X (2005) High-performance, power-aware distributed computing for scientific applications. In: IEEE computer, vol. 38, 2005
9. Rodero I, Chandra S, Parashar M, Muralidhar R, Seshadri H, Poole S (2010) Investigating the potential of application-centric aggressive power management for HPC workloads. In: Proceedings of IEEE conference on high performance computing, HiPC 2010
10. Etinski M, Corbalan J, Labarta J, Valero M, Veidenbaum A (2009) Power-Aware load balancing of large scale MPI applications. In: Proceedings of IEEE symposium on parallel and distributed processing, IPDPS 2009, doi: 10.1109/IPDPS.2009.5160973
11. Ma Y, Gong B, Zou L (2009) Energy-Efficient Scheduling Algorithm of Task Dependent Graph on DVS-Unable Cluster System. In: Proceedings of IEEE/ACM conference grid computing, doi:10/1109/GRID.2009.5353056
12. Li D, Nikolopoulos DS, Cameron K, Supinski BR, Schulz M (2010) Power-aware MPI Task Aggregation Prediction for High-End Computing Systems. In: Proceedings of IEEE symposium parallel and distributed processing symposium, IPDPS, 2010
13. Dong Y, Chen J, Yang X, Yang CY, Peng L (2008) Low Power Optimization for MPI Collective Operations. In: Proceedings of IEEE conference young computer scientists, ICYCS 2008, doi: 10.1109/ICYCS.2008.500
14. Y. Chen and Y. Zeng (2011) "Automatic energy status controlling with dynamic voltage scaling in power aware high performance computing cluster". Proc. IEEE Conf. Parallel and distributed computing, applications and technologies (PDCAT), Oct. 2011, pp. 412–416, doi: 10.1109/PDCAT.2011.24
15. Kim NS, Austin T et al (2003) Leakage current: Moore's law meets static power. IEEE Computer, vol. 36, doi:10.1109/MC.2003.1250885

16. Ahmad HF (2002) Multi-agent systems: overview of a new paradigm for distributed systems. In Proceedings of IEEE symposium high assurance systems engineering
17. Venkitesh S, Bindhumadhava BS, Bhandari AA (2006) Implementation of automated grid software management tool: a mobile agent based approach. In Proceedings of international conference on information and knowledge engineering
18. MPICH2-1.4.1. http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-1.4.1-userguide.pdf