A High Performance Cluster System Design by Adaptive Power Control*

Masaaki Kondo, Yoshimichi Ikeda, Hiroshi Nakamura
Research Center for Advanced Science and Technology,
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, Japan
{kondo,ikeda,nakamura}@hal.rcast.u-tokyo.ac.jp

Abstract

The first order design constraint in dense packaged clusters is power consumption. The currently developed cluster systems are conservatively designed so that the expected peak power does not exceed the power limit. However, practical power consumption seldom reaches the peak power. In this paper, we propose a new approach to design a high performance cluster system by an adaptive power control technique. Our approach is to integrate many computation nodes into a system whose total theoretical peak power exceeds the limit and to control runtime effective power by optimizing the number of working nodes and/or the clock frequency of the processors. We show the algorithm of the adaptive power control and performance evaluation by using a real cluster system. Evaluation results show that our proposed approach greatly improves performance as large as 46% compared to a conventional cluster system.

1. Introduction

Compact and dense packaging is indispensable for cluster systems to achieve high performance/cost ratio since the required installation space is one of the most cost sensitive intensive aspects in the cluster system design. The first order design constraint of dense packaging is power consumption and cooling capability of a system. The peak power consumption of a system must be below the maximum power budget determined by the power delivery or the cooling subsystems. Therefore, power awareness has become an important issue for high performance clusters and has been focused in the HPC community for recent years.

*This work was supported in part by Japan Science and Technology Agency as a CREST research program entitled "Mega-Scale Computing Based on Low-Power Technology and Workload Modeling" and by Toray Science and Technology Grant.

As a result, the recent trend of high-performance clusters is using a relatively low-power processor for a node and densely mounting many of these nodes. Several high performance cluster systems have been developed with this design style. These include Green Destiny [23], BlueGene/L [3], Orion Multisystem machines [2], and MegaProto [19]. Their goal is to achieve higher performance by large-scale parallel processing with many power-efficient processors instead of using fewer powerful high-end processors.

The currently developed cluster systems are conservatively designed so that the expected peak power does not exceed the limitation of the power consumption. Programs are executed without any concern about power or thermal limitation when the systems are in appropriate operating conditions. However, practical power consumption seldom reaches the peak power of a system. Because the activity of each component of systems highly depends on the behavior of executing programs and the peak activity does not usually continue, effective power consumption is often below the peak power. The difference between the peak power and the effective power consumption can be considered as a *surplus* of power. If this surplus of power budget is used for increasing the number of cluster nodes or increasing the clock speed of processors, we can possibly boost the performance.

In this paper, we propose a new approach to design a high performance cluster system by an adaptive power control technique. We integrate many computation nodes into a system whose total theoretical peak power exceeds the power limitation determined by cooling systems. Although power consumption is possibly beyond the limitation, effective power is controlled to be below the limitation by adaptively managing the system configuration. This control of the effective power is realized by adjusting the number of working nodes and the clock frequency of the processors on the working nodes. Because power consumed in the cluster is reduced using only subset of the nodes and/or using lower clock frequency, power constraint can be maintained even in programs which have high processor utilization.

The adaptive power control consists of two steps. One is optimizing the number of working nodes statically and the other is controlling processors' clock frequency dynamically. We use performance and power profiles to find the optimal number of nodes in terms of performance. The application is executed using the selected number of nodes. DVFS (Dynamic Voltage and Frequency Scaling) technique, which is applied especially in mobile computing environment, is used to control the power at runtime. We use a runtime power monitoring framework and a feedback power control method.

Several researchers have focused on saving energy in HPC cluster systems [10, 9, 16, 13] with DVFS. Their goal is to save energy/power consumption with little performance impact. They try to find program regions where the performance does not degrade much even if clock frequency of processors is lowered. On the other hand, our goal is to enhance performance of cluster systems with DVFS under the power (NOT energy) limitation. Although saving energy/power is important, there is still a great demand for higher performance. Users in supercomputing centers usually have time limitation for their use of computational resources. Higher execution throughput (performance) is one of the most critical issues in such a situation. Moreover, the cost of air conditioning is very high in hosting centers. Shorter execution time greatly contributes to reducing the air conditioning cost per task. These points motivate us to propose a new high performance cluster system design and an adaptive power control method.

This paper is organized as follows. The next section describes the related work. In Section 3, we show the proposed cluster system design and algorithm for optimizing the number of nodes and clock speed. Section 4 describes the experimental environment and assumptions. The evaluation results are presented in Section 5. Finally, we conclude in Section 6.

2. Related Work

The power and energy characteristics of a high performance cluster system are analyzed in [7] with power-performance profiles of the NAS parallel benchmarks. The work in [12] studies feasibility of using DVFS to reduce the thermal power and energy in DVFS-enabled Beowulf clusters and shows that a significant amount of system power and energy is saved with little performance degradation. The tradeoff between energy and performance is studied in [8] in a DVFS-enabled cluster. In this research, the effect of changing the number of working nodes is evaluated in addition to the DVFS technique.

The various approaches are proposed to reduce power and energy consumption with negligible performance impact for high performance clusters by static program analysis and profiling [10, 17, 9]. Kappiaho et al. have proposed *Jitter* [16], which reduces power consumption in the presence of load imbalance. They change clock speed of processors on nodes that are assigned less computation. The work in [22] tries to find an optimal schedule (number of nodes and CPU frequency) that simultaneously satisfies an energy limit and minimizes execution time for a given upper limit of energy consumption. A runtime algorithm that automatically and transparently adapts voltage and frequency settings of processors is proposed in [13].

Felter et al. have proposed *power shifting* [6] which tries to maximize performance under reduced fixed power budgets. They dynamically allocate power among components of server systems (processor and memory subsystem).

Our proposal is a cluster system design where peak power potentially exceeds the power limitation, and then the number of nodes and clock speed is controlled so that the effective power is still below the limitation. With this design, our aim is to boost performance by finding optimal operating settings. In these points, our work is different from other studies.

The work by Li and Martinez [18] tries to optimizes power consumption of a parallel application on a chip multiprocessor (CMP) by changing the number of active processors and clock speed dynamically under a given performance constraints. Although their concept of is similar to our proposal, their work targets shared memory systems. In those systems, the number of processors can be changed easily compared to distributed memory systems. Therefore, their method cannot be extended straightforwardly for cluster systems, which are usually distributed memory systems.

Rubio et al. have proposed *Dynamic Processor Overclocking (DPO)* which boosts the processor clock frequency to values higher than the nominal whenever the power/cooling constraints are not violated [21]. They analyzed the benefit of DPO through experiment on a real machine and identified the application characteristics that determine the extent of performance improvement. Although the concept of DPO is similar to ours, They focused on the single processor system performance and did not consider the parallel computation environment.

3. Cluster System Design with Adaptive Power Control

3.1. Concept of Proposed Cluster System Design

The currently developed cluster systems are conservatively designed so that the expected peak power does not exceed the limitation of the power consumption. Let P_{peak} and P_{limit} be the peak power for a node (usually written in catalog specifications) and power limit determined by cooling capability. Suppose the total number of nodes is N_{total}. The conventional cluster is designed such that $P_{limit} \geq P_{peak} \times N_{total}$.

In many computer systems, a large portion of power is dissipated in a processor or sometimes in a memory subsystem. Since the activity of a processor or a memory system highly depends on the characteristics of executing programs, effective power consumption of a system is different for different programs or different phases within a program. This diversity of power consumption mainly comes from the occurrence of cache misses or I/O activities [19, 7]. When an executing program exhibits a high cache miss rate, the power consumption of the processor decreases while that of the memory system increases. Similarly, the effective power consumption is influenced by the ratio of computation to communication in parallel programs.

Many studies have explored runtime power characteristics and found that the average power consumption often below the peak power [19, 6, 15]. The difference between the peak power and the runtime effective power consumption can be considered as a *surplus* of power. If this surplus of power is used for increasing the number of cluster nodes or increasing the clock speed of the processors, we can possibly boost the performance. To this end, we propose a new cluster system design with adaptive power control.

In the proposed cluster system, P_{peak} is allowed to exceed the P_{limit}. When P_{idle} is power consumption when a node is idle, the possible maximum organization of our cluster system design is implementing a total N_{total} nodes such that $P_{limit} < P_{peak} \times N_{total}$ & $P_{limit} \geq P_{idle} \times N_{total}$.

Let $P_{effective}$ be the runtime effective power consumption of a node. When all the nodes participate in a computation and clock frequency of processors is high, $P_{effective}$ can exceed the limit ($P_{limit} < P_{effective} \times N_{total}$). However, by controlling number of working nodes (N_{used}) and clock frequency, we can still maintain effective power consumption within the limit ($P_{limit} \geq P'_{effective} \times N_{used}$).

Because the best configuration (the number of working nodes and clock frequency) is program dependent, we should adaptively control the configuration for a given application. The policy of the adaptive power control is controlling the configuration to satisfy the following equation: $P_{limit} \approx P_{effective} \times N_{used}$. The next section describes how to optimize the number of nodes and clock frequency of the processors for our cluster system design.

3.2. Adaptive Power Control

3.2.1 Overview

The problem we address is optimizing the cluster configuration, that is, the number of working nodes used for a computation and clock frequency of the processors, adaptively for a given power limit and for a given program.

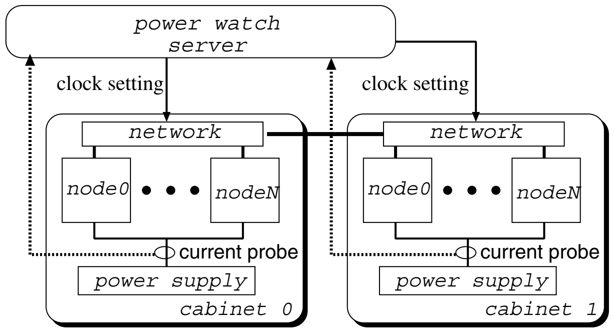

Figure 1. Runtime power control

The optimal configuration may be different for different phases in a program. Therefore, the adaptation should be deployed dynamically. However, changing the number of nodes during program execution is not practical because of its large overhead. Since we suppose target applications are written by the distributed memory programming style, a dataset is usually divided and assigned into each node. Increasing or decreasing number of nodes requires redistribution of data. Moreover, it greatly complicates parallel programming. Therefore, fine-grain effective power control is done by DVFS with monitoring the runtime power consumption. This allows responding to dynamical behavior changes of a program execution. Since the cost of changing clock frequency of processors is relatively small, this is feasible. The number of working nodes is determined statically by profiling.

In dense packaged clusters, several nodes are integrated into a cabinet which has its own power supply. For example, MegaProto [19] integrates 16 Transmeta efficion nodes into a 1U cabinet which is mountable in a standard 19-inch rack. Another example is commercial blade servers where dozens of nodes are integrated into a 4U – 6U high cabinet. The power limit is usually defined for each cabinet because the thermal constraint derived by cooling capability is the property of a cabinet. Therefore, the power limit of each cabinet should be taken into account to decide a optimal configuration.

3.2.2 Runtime Clock Control Method

The goal of our runtime clock control (RCC) is to set the clock frequency of the processors as high as possible within a given power constraint. The clock frequency is changed based on the current power usage of the system which is monitored by an external node called *Power Watch Server* via a current probe as shown in Figure 1. Power consumption is measured and clock setting is determined for each cabinet because the power limit is the property of a cabinet

```
$C_{overshoot} = 100 \,/\, Th_{overshoot};$

$while(1)$ {
  /* for every power measurement cycle */
  $P_{now} = get\_power();$

  $if\ (P_{now} \geq P_{limit})${
    $Down\_Clock\_Speed();$
    $T_{last\_overshoot} = T_{now};$
  }
  $elseif\ (T_{now}\ \%\ T_{itvl} == 0)${
    $if\ (T_{now} - T_{last\_overshoot} > C_{overshoot})${
      $Up\_Clock\_Speed();$
    }
  }

  $T_{now}$++;
  $Sleep(T_{sampling});$
}
```

Figure 2. Algorithm of Runtime Clock Control

as mentioned earlier. The clock speed of all the processors in a cabinet is uniformly controlled.

We permit effective power consumption to overshoot the power limit for a very short period of time. Because infrequent power spikes will not affect the system temperature and usually a power-supply is designed to tolerate short time spikes, this does not cause a reliability problem. We introduce a user specified threshold denoted as $Th_{overshoot}$ which indicates the target percentage of time that effective power is permitted to overshoot the power limit, relative to total execution time.

The Figure 2 presents the algorithm of RCC which is implemented in the Power Watch Server. In the figure, we suppose the sampling cycle time of power measurement is $T_{sampling}$ second which is limited by the time resolution of the power measurement tool. A smaller sampling cycle is preferable to follow sudden changes of execution behavior.

For every $T_{sampling}$, the current power consumption (P_{now}) is monitored. If it is beyond the power limit, the server sends a "clock down" message to all the working nodes and the current time is recorded on $T_{last\_overshoot}$. On the other hand, the server tries to increase the clock frequency for every time interval of T_{itvl}. The purpose of T_{itvl} is to absorb the effect of power fluctuation. If the period of $C_{overshoot}$ is passed since the last overshot time, the server actually sends a "clock up" message to all the nodes. This almost ensures that the ratio of the total overshot period to total execution time is within $Th_{overshoot}$.

3.2.3 Determining the Number of Working Nodes

In order to determine the number of working nodes, we use profiling. There are many profiling methods to enhance performance or to save energy consumption. Basically method for selecting the best number of nodes to be used in the execution (denoted as N_{opt}) is orthogonal to the way of profiling. Several kinds of profiling methods can be applied for this purpose. In this paper, we use a test-run based approach.

A short test-run which consists of few iterations of a target program is repeatedly executed varying number of nodes. This is feasible to estimate an overall power-performance trend for a power constrained cluster system since the power-performance characteristic is fairly regular throughout program iterations in scientific applications [7]. More specifically, we obtain MIPS (Million Instruction per Second) for each test-run. We skip a first few second (T_{skip}) from the beginning of the program execution and then evaluate MIPS using performance counters for a certain period of time (T_{prof}). The RCC method described in the previous subsection is applied to profiling to maintain the effective power consumption below the power constraint.

As power limit is the property of a cabinet, N_{opt} should be determined for each cabinet. Here, we assume the computational load is evenly distributed among cluster nodes since our target is scientific applications. Profiling is done with only one cabinet and we chose N_{opt} for that cabinet. When a system consist of several cabinets, the obtained N_{opt} value is applied for all the cabinets. For example, when a cabinet has 8 nodes and a system consists of the 4 cabinets and if N_{opt} is found to be 6 nodes, the total number of nodes to be used is 24.

The following is the procedure of our profiler.

1. Execute a target program with N-nodes (initial value of N is the maximum number nodes in a cabinet).

2. After skipping T_{skip}-second, evaluate performance (MIPS) for T_{prof}-second.

3. Terminate the program and recode the obtained MIPS.

4. Decrement N by one and go back to step 1 until the maximum clock frequency is always used in the RCC method throughout the execution in step 2.

5. Chose the number of nodes in which the best MIPS is obtained.

Because the number of nodes in one cabinet is not large (e.g. at most 16 nodes), the required time for the above profiling procedure is short. Therefore, the cost of the profiling is very low even in large scale cluster systems.

Table 1. Specification of a node

M/B	Commell LV673 [5] - i915GM + ICH6M chip-set - Gb Ethernet x 2
Processor	Pentium M 760 [14] (Max 2GHz, FSB533MHz)
Memory	DDR2-SDRAM 1GB

Table 2. Clock and voltage setting and peak power of a node

Clock (GHz)	Core Vdd (V)	Peak Power(W)
2.00	1.356	56.6
1.86	1.308	52.9
1.73	1.260	50.0
1.60	1.228	48.1
1.46	1.196	45.3
1.33	1.164	43.6
1.20	1.132	42.0
1.06	1.084	41.9
0.80	0.988	39.1

4. Experimental Environment

4.1. Cluster System Specification

We evaluated the proposed cluster system in a real experiment. We used an Intel Pentium M-based PC for a cluster node running Linux kernel-2.6.11. One cabinet has 8 nodes ($N_{total} = 8$) which are connected by a Gigabit-ethernet network. Table 1 shows the specification of the node. Table 2 presents available clock frequency and voltage settings for the processor. The clock frequency of the processor is controlled by the *cpufreq* interface.

The NAS parallel benchmark [4], HPL [20], and Himeno-bench [1] are used for the evaluation. Note that the HPL, Himeno-bench, and EP benchmark can be executed on a flexible number of nodes, while the other NBP programs are executed only on 4 or 8-node configuration. All the applications were compiled with gcc.

4.2. Power Measurement Environment

We measured runtime power consumption by an electric current measurement tool with Hall elements [11]. This tool has ringshape Hall element probes through which power lines to be measured without any electrical contacts nor interferences. The tool also has A/D converters to digitize measured power currents and to transfer them to the Power Watch Server with a fine-grained time resolution as fine as 10ms. Note that we pessimistically suppose the resolution is 20ms. Thus, we use $T_{sampling}$ of 20ms for the RCC algorithm in the evaluation.

Using this tool, we measured the power current of +12V DC outputs of the power supply which are input to all of the cluster nodes.

Table 3. Possible maximum clock frequency for conventional clusters

# of nodes	Possible max. clock (GHz)	
	200W limit	250W limit
3	2.00	2.00
4	1.73	2.00
5	0.8	1.73
6	N/A	0.8
7	N/A	N/A
8	N/A	N/A

4.3. Peak Power of the System

We first evaluated the peak power consumption of one node. The High-Performance Linpack (HPL) code [20] is used for estimating the peak power because a highly optimized version of the Linpack code is assumed to exhibit worst case peak power [21]. Note that this estimation is done with a single node.

Table 2 shows the peak power consumption of the node used in the evaluation for each clock frequency setting. Since the worst case peak power must be below the power limit in conventional cluster systems, the allowable clock speed which is used for computation is restricted. Table 3 shows allowable clock frequency for one cabinet on the conventional cluster system for various number of nodes when the power limit is supposed to be 200W and 250W.

5. Experimental Result

We evaluate the performance of the proposed cluster compared with a conventional cluster under the following assumptions.

- $P_{limit} = 200W$ or $250W$.
- Parameters for RCC
  - $Th_{overshoot} = 1\%$.
  - $T_{sampling}$ = 20[ms].
  - T_{itvl} = 10.
- Parameters for the profiler.
  - T_{skip} = 15[s].
  - T_{prof} = 10[s].

Table 4. Chosen number of nodes (and clock frequency)

	200W power limit		250W power limit	
	Proposed	Conventional	Proposed	Conventional
HPL	4-node	4-node,1.73GHz	6-node	4-node,2.00GHz
Himeno	6-node	4-node,1.73GHz	6-node	5-node,1.73GHz
EP	5-node	4-node,1.73GHz	6-node	5-node,1.73GHz
CG	4-node	4-node,1.73GHz	8-node	4-node,2.00GHz
FT	4-node	4-node,1.73GHz	8-node	4-node,2.00GHz
IS	4-node	4-node,1.73GHz	8-node	4-node,2.00GHz
LU	4-node	4-node,1.73GHz	8-node	4-node,2.00GHz
MG	4-node	4-node,1.73GHz	8-node	4-node,2.00GHz

5.1. Performance Benefit

Table 4 presents the number of working nodes selected by profiling for the proposed cluster. The configurations of conventional cluster are chosen from Table 3 and the best case are presented in the table.

Figure 3 and Figure 4 show performance of the proposed cluster normalized by that of conventional cluster using one cabinet (total 8-nodes) and two cabinets (total 16-nodes), respectively.

First, we discuss the result of one cabinet. As shown in Figure 3, the proposed cluster achieves higher performance than the conventional cluster in all the evaluated programs. Especially in MG in 250W power limit case, 46% of higher performance is obtained. Averaging across all the benchmarks, the speedup reaches 8% and 31% in 200W and 250W power limit, respectively.

One reason for this result is that more number of working nodes is utilized for the computation in the proposed cluster. As shown in Table 4, the proposed cluster uses more number of nodes compared to the conventional cluster in two programs in 200W power limit (Himeno-bench and EP) and all the programs in 250W power limit. In conventional cluster, though the effective power is far below the power limit, the number of working nodes is restricted by the worst case peak power. On the other hand, the proposed cluster is restricted by just effective power consumption, and therefore the surplus of power is utilized for additional computation power.

Another reason why the proposed cluster is superior to the conventional cluster is that the proposed cluster uses higher clock speed than the conventional cluster. The clock frequency of the processors in conventional cluster is fixed to guarantee runtime effective power never exceeds the power limit. Since the proposed cluster controls the clock frequency dynamically by monitoring the effective power consumption with the RCC algorithm, higher clock frequency can be safely used. In the case of 200W power limit, the proposed and conventional cluster use the same number of working nodes in most of the programs. In that cases, performance benefit of proposed cluster comes only from RCC. This results indicates that the RCC method is very effective for boosting performance for power constrained cluster systems.

Figure 5 shows the runtime power profile for HPL benchmark for both proposed and conventional clusters in the case of 250W power limit. As seen from the figure, effective power of conventional cluster is far below the power limit. Because communications are essentially required in parallel programs, the activity of the processors becomes low even in HPL (Linpack) benchmark. On the other hand, the power profile of proposed cluster is likely to be just within the limit.

Second, we discuss the result of two cabinets shown in Figure 4. As seen from the figure, the proposed cluster achieves higher performance than the conventional cluster in all the programs except for LU in 250W limit. Performance improvement for each program is similar to that in the result for one cabinet shown in Figure 3. This indicate that the number of working node derived from profiling in which only one-cabinet is used for test-run is suitable to be applied to all the cabinet in the system.

Performance improvement becomes slightly worse in CG and MG in 250W limit compared with the one cabinet case. Moreover, performance gets worse than the conventional cluster in LU. Because the number of nodes to be used is limited in these programs (the number of nodes should be a power of two), a suitable range of clock frequency cannot be selected even by RCC in the assumed 250W power limit. Especially in LU, the program is executed with the lowest clock frequency in most of the execution time in the proposed cluster.

5.2. Profiling Accuracy

To evaluate the profiler efficiency, we measure performance (MIPS) of HPL, Himeno-bench, and NPB EP for several profiling sampling periods ($T_{prof} = 1, 5, 10$) varying the number of working nodes. As mentioned earlier,

Table 5. Selected number of nodes by profiling (performance degradation from the best case)

	200W limit				250W limit			
	T_{prof}=1	T_{prof}=5	T_{prof}=10	full	T_{prof}=1	T_{prof}=5	T_{prof}=10	full
HPL	5-node (11%)	4-node (0%)	4-node (0%)	4-node	6-node (0%)	6-node (0%)	6-node (0%)	6-node
Himeno	5-node (0%)	6-node (0.2%)	6-node (0.2%)	5-node	6-node (0%)	6-node (0%)	6-node (0%)	6-node
EP	6-node (10%)	5-node (0%)	5-node (0%)	5-node	8-node (6.9%)	6-node (0.1%)	6-node (0.1%)	7-node

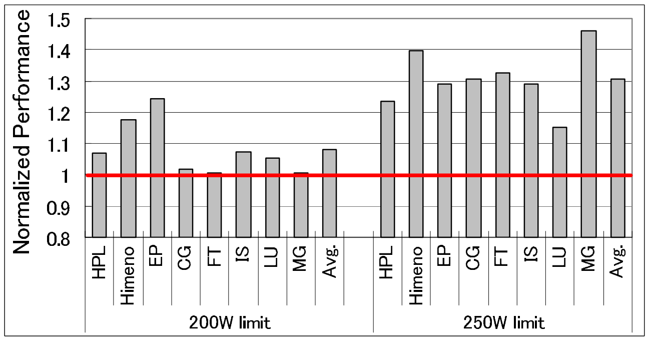

Figure 3. Performance of proposed cluster (1-cabinet)

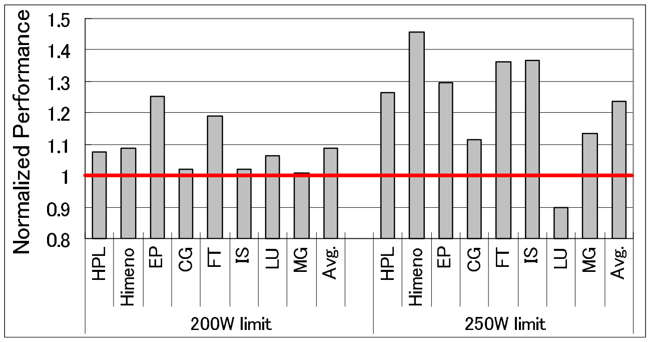

Figure 4. Performance of proposed cluster (2-cabinets)

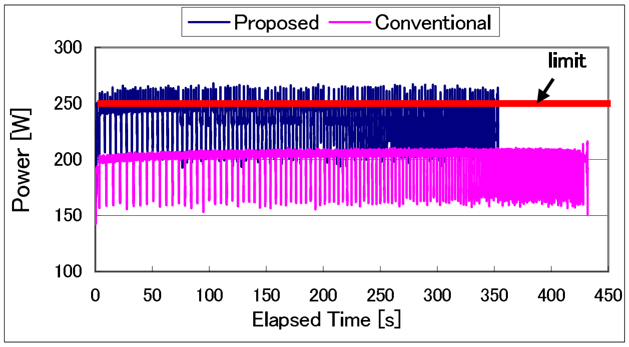

Figure 5. Power profile for HPL

we use just one cabinet for profiling. Thus, the maximum number of nodes is 8.

Table 5 shows the selected number of nodes which exhibits the highest MIPS value for each sampling period. In the table, (*full*) represents the actual best case when the program is executed from begging to completion. The table also shows the percentages of performance degradation compared with the best case when the selected number of nodes is used for execution.

As seen from the table, when T_{prof} is 1-second, the selected number of nodes is different from the best one written in *full* columns in several cases, and thus, obtained performance is very low compared with the best cases. However, if we use T_{prof} of 5 or 10-seconds, our profiling technique chooses the best number of nodes in most of the cases. Although the different number of nodes is selected for Himeno-bench in 200W and EP in 250W limit, performance degradation is negligible (below 1%). Since the RCC method tunes the frequency of the processors, the performance penalty caused by selecting un-optimal number of nodes is alleviated. This indicates that the short test-run based profiling method is very effective for selecting number of working nodes for the proposed cluster system.

5.3. Parameter Sensitivity

In the previous evaluation, the target overshot ratio ($Th_{overshoot}$) was set to 1%. To evaluate how this parameter affects system performance, we evaluate the proposed cluster varying the value of $Th_{overshoot}$. Figure 6 presents the normalized performance of the proposed cluster for three cases of $Th_{overshoot}$ values, 1%, 5% and 10% with one cabinet under the 250W power limit.

As shown in the figure, a slightly higher performance is observed with larger $Th_{overshoot}$ values. This is because the number of "attacks" (issue of "clock up") increases and thereby, the opportunity of using higher clock frequency increases. However, the difference of performance is not sig-

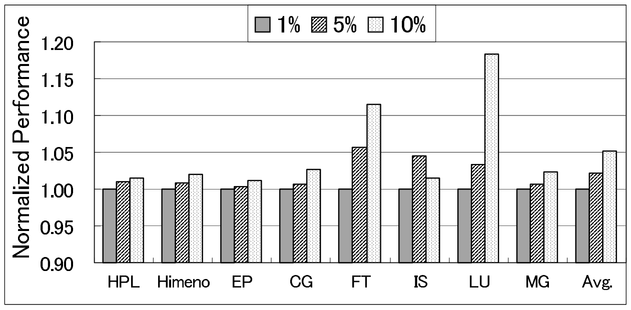

Figure 6. Results varying $Th_{overshoot}$

nificant in most of the programs. Even in $Th_{overshoot}$ of 10%, performance improvement is 5% on average. Since the total overshot period becomes long if we use a larger $Th_{overshoot}$ value, a small $Th_{overshoot}$ is preferable for the view point of reliability. The parameter of $Th_{overshoot}$ should be determined with considering the performance benefit and the reliability. We believe the target overshot ratio of 1% is fairly balanced point.

6. Concluding Remarks

In this paper, we proposed a new cluster system design with adaptive power control. The idea behind the proposed cluster system is integrating many working nodes into a system whose total theoretical peak power exceeds the power limitation determined by cooling systems. Although power consumption is possibly beyond the limitation, the effective power is controlled to be below the limitation by adaptively managing the system configuration, that is, the number of working nodes and clock frequency of the processors. We also showed algorithm for optimizing configurations.

We evaluated the performance improvement derived from the proposed cluster system by a real PC clusters. The evaluation results revealed that the proposed cluster achieved higher performance than a conventional cluster. Our proposed cluster system fully utilizes the available power budget without any reliability issue. From these results, we can conclude that the proposed cluster design is superior to conventional cluster designs.

We are planing to evaluate our proposed concept with larger scale cluster systems. We are also interested in thermal management technique for our cluster systems.

References

[1] Himeno benchmark. http://w3cic.riken.go.jp/ E/HPC_e/ HimenoBMT_e/oldver_e.htm.

[2] Orion multisystems. http://www.orionmulti.com/.

[3] N. Adiga and et al. An overview of the bluegene/l supercomputer. In *Supercomputing 2002*, Nov. 2002.

[4] D. Bailey and et al. The nas parallel benchmarks 2.0. In *NASA Ames Research Center Report, NAS-05-02*.

[5] COMMELL. Mini-itx express motherboard lv-673 datasheet. http://www.commell.com.tw/ Product/SBC/LV-673.HTM.

[6] W. Felter and et al. A performance-conserving approach for reducing peak power consumption in server systems. In *ICS 2005*, pages 293–302, June 2005.

[7] X. Feng, R. Ge, and K. Cameron. Power and energy profiling of scientific applications on distributed. In *IPDPS 2005*, Apr. 2005.

[8] V. Freeh and et al. Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In *IPDPS 2005*, Apr. 2005.

[9] V. Freeh and et al. Using multiple energy gears in mpi programs on a power-scalable cluster. In *PPoPP 2005*, pages 164–173, June 2005.

[10] R. Ge, X. Feng, and K. Cameron. Improvement of power performance efficiency for high-end computing. In *Workshop on HP-PAC 2005*, Apr. 2005.

[11] Y. Hotta and et al. Measurement and characterization of power consumption of microprocesors for power-aware cluster. In *COOL Chips VII*, Apr. 2004.

[12] C. Hsu and W. Feng. A feasibility analysis of power awareness in commodity-based high-performance clusters. In *Cluster 2005*, Sept. 2005.

[13] C. Hsu and W. Feng. A power-aware run-time system for high-performance computing. In *Supercomputing 2005*, Nov. 2005.

[14] Intel. Pentium m processor on 90nm process with 2-mb l2 cache datasheet, Jan. 2005.

[15] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *the 36th MICRO*, pages 93–104, Dec. 2003.

[16] N. Kappiah, V. Freeh, and D. Lowenthal. Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs. In *Supercomputing 2005*, Nov. 2005.

[17] R. Kotla and et al. Scheduling processor voltage and frequency in server and cluster systems. In *Proc. IPDPS 2005*, Apr. 2005.

[18] J. Li and J. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *the 12th HPCA*, pages 77–87, Feb. 2006.

[19] H. Nakashima and et al. Megaproto: 1tflops/10kw rack is feasible even with only commodity technology. In *Supercomputing 2005*, Nov. 2005.

[20] A. Petitet and et al. Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers.

[21] J. Rubio and et al. Dynamic processor overclocking for improving performance of power-constrainded systems. In *IBM Research Report RC23666(W0507-124)*.

[22] R. Springer and et al. Minimizing execution time in mpi programs on an energy-constrained, power-scalable cluster. In *PPoPP 2006*, pages 230–238, Mar. 2006.

[23] M. Warren and et al. High density computing: A 240-node beowulf in one cubic meter. In *Supercomputing 2002*, Nov. 2002.