

Profile-based Optimization of Power Performance by using Dynamic Voltage Scaling on a PC cluster

Yoshihiko Hotta[†], Mitsuhisa Sato[†], Hideaki Kimura[†],
Satoshi Matsuoka[‡], Taisuke Boku[†], Daisuke Takahashi[†]

[†] Graduate School of Systems & Information Engineering, University of Tsukuba
{hotta,msato,kimura,taisuke,daisuke}@hpcs.cs.tsukuba.ac.jp
[‡] Tokyo Institute of Technology {matsu@titech.is.ac.jp}

Abstract

Currently, several of the high performance processors used in a PC cluster have a DVFS (Dynamic Voltage and Frequency Scaling) architecture that can dynamically scale processor voltage and frequency. Adaptive scheduling of the voltage and frequency enables us to reduce power dissipation without a performance slowdown during communication and memory access. In this paper, we propose a method of profiled-based power-performance optimization by DVFS scheduling in a high-performance PC cluster. We divide the program execution into several regions and select the best gear for power efficiency. Selecting the best gear is not straightforward since the overhead of DVFS transition is not free. We propose an optimization algorithm to select a gear using the execution and power profile by taking the transition overhead into account. We have built and designed a power-profiling system, PowerWatch. With this system we examined the effectiveness of our optimization algorithm on two types of power-scalable clusters (Crusoe and Turion). According to the results of benchmark tests, we achieved almost 40% reduction in terms of EDP (energy-delay product) without performance impact (less than 5%) compared to results using the standard clock frequency.

1 Introduction

Recently, there has been tremendous interest in power-aware computing for mobile embedded systems such as PDAs and cellular phones. Even in high-end parallel computing systems, it is a very important issue to reduce the power consumption for cooling and high-density packaging. This is also a serious problem in the servers of data centers. For example, a Google engineer has warned that “Performance does not matter. Power is the most important matter for managing our systems.” In the past decade, processor performance has rapidly increased while power dissipation has increased.

The explosive increase of processor power consumption has had a big impact on the design of large-scale systems

such as the Earth Simulator and ASCI machines. One solution to the problem of power consumption is found in BlueGene/L [1]. It utilizes low-power components from commodity technology to reduce the power consumption of the entire system. The processor used in BlueGene/L is an embedded, customized PowerPC chip-multiprocessor. High-performance, low-power clusters such as Green Destiny [10] have been built with low-power processors with high-density packaging. Using Transmeta Efficeon processors we have been developing a low-power and high-density cluster called MegaProto [6] to achieve “mega-scale computing,” which is a large-scale computing platform with thousands of processors.

Dynamic voltage scaling (DVS) is recognized as one of the most effective power-reduction techniques. As a major portion of the power of CMOS circuitry scales quadratically with the supply voltage, lowering the supply voltage can significantly reduce power dissipation. Modern low-power processors such Intel Pentium-M and Transmeta Crusoe have the DVS mechanism to change the frequency scaling with the voltage, which usually allows longer battery life in mobile platforms. Even high-performance processors used for HPC (high-performance computing) PC clusters also have DVS. DVS may be used to reduce the power consumption by changing the clock frequency-voltage setting without impacting the time-to-solution.

In this paper, we propose a method of profile-based power-performance optimization by using DVS in a high-performance PC cluster. In parallel applications, a large amount of execution time may be spent for communications to exchange data between nodes in a cluster. When a node waits for communication from other nodes, we can reduce the power dissipation by setting a lower clock frequency of the processor. On the other hand, it is often found that increasing the CPU clock frequency does not always result in an increase of performance because memory is sometimes the bottleneck in memory-intensive applications. In

this case, lowering the clock frequency can save power dissipation with less performance penalty.

The DVS of a particular processor defines the possible combinations of clock frequency and voltage which we call *gear*. We divide the program execution into several regions and choose the best gear to execute each region according to the profile information on the execution timing and power of the previous trial run. Choosing the best gear for each region is not straightforward since the overhead of the transition from one gear to a different gear is not free. It may take tens of microseconds to switch from one gear to another so that the overhead may eliminate the benefits of DVS. Our proposed algorithm to choose the best gear takes this overhead of transition into account to optimize the power performance. Selecting regions is another important issue. In this paper, we set the regions manually for the communications and memory-intensive regions, and used these region settings to evaluate our proposed method.

LongRun [4], developed by Transmeta, is an interesting facility that automatically controls DVS at the micro-architecture level. Although it is very useful for a laptop computer with long battery life, we found that it may fail to select the best gear for HPC applications [6]. We believe that exploiting application knowledge by using the execution profile is more effective in achieving better power performance in high-performance parallel applications.

The metric for measuring power performance is also a very important and interesting issue. In high-performance systems, minimizing the execution time is usually a first priority. Even if the total energy is minimized, the user would not be satisfied with poor performance. We choose the energy-delay product (EDP) as a metric to optimize power performance by taking the execution time into account.

In this paper, we first present the power-performance characteristics of parallel programs at several voltage and frequency settings. For our experiments, we built a power-profile system named PowerWatch using the Hall device, which allows us to measure the power consumption of each node at as much as a hundred-microsecond resolution without changing hardware. In [7], we reported the power-performance characteristics of various processors such as Crusoe, XScale, and Pentium by using this system. We used the NAS benchmark suite for evaluation. We also built two kinds of clusters using low-power processors for the experiment: the Turion cluster and Crusoe cluster. Our results shows our optimization using the profile is promising for an HPC power-scalable cluster.

The rest of this paper is organized as follows. We list the related work in the next section. Our method of profiled-based power-performance optimization is described in Section 3. Section 4 presents the experiment setup and results of our method using NAS parallel benchmarks. The discussions of our method and power-performance efficiency is

presented in Section 5. Finally, we present our concluding remarks and future work in Section 6.

2 Related Work

There has been tremendous interest in power-aware computing in mobile and embedded systems. Many researchers have tried to reduce energy consumption to enable these systems to have a long battery life. For example, a laptop computer has a simple method that reduces the frequency and voltage when the power supply is inactive.

Recently, there have been several case studies using DVFS to reduce power and energy consumption even in high-performance computing.

Rong [5] presented a profiled-based optimization using the MPE (Multi-Processing Environment) tool included in MPICH. They proposed DVFS scheduling for distributed power-aware HPC clusters. By varying the scheduling granularity, they achieved high energy saving (36% without a performance decline). And they also evaluated the EDP to automatically select a distributed DVFS that meets the performance demand. While their approach is similar to ours, their profiler can focus only on the communication regions. Our method can be applied to optimize not only communication regions but also other regions.

Kappiah [9] presented a system called Jitter (Just-In-Time DVFS). It reduces the clock frequency on nodes which have been assigned small computations. In such a case, it has slack time to wait for other nodes which have been assigned large computation. This saves energy on the nodes. They focused on iterative programs, which comprise the vast majority of scientific programs, since the iterations of these programs are stable; consequently, they were able to estimate future behavior according to past behavior. By manually inserting special code for the system, they achieved an energy reduction of 8% while execution time was reduced by 2.6%.

Run-time solutions of DVFS have also been investigated. Hsu [8] proposed a power-aware DVFS run-time system that automatically adapts a large power reduction with small performance loss. They use β adaptation, which is scheduling by a β value that can be used to predict the behavior of the program. And, they proposed a model based on the MIPS rate.

Our work differs by using fine-grained profile information to define the best *gear* in each region in high performance distributed system in terms of power performance. We propose an algorithm which include transition cost of DVS to define the *gear*, and estimate the result as two types of metric. Additionally it enables us easily to use optimization in all kinds of application.

3 Profiled-based DVS Optimization

3.1 Metric of Power Performance

There have been several metrics for high-performance computing such as FLOPS (floating-point operations per

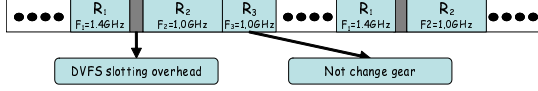


Figure 1. Regions and overhead of a DVS transition

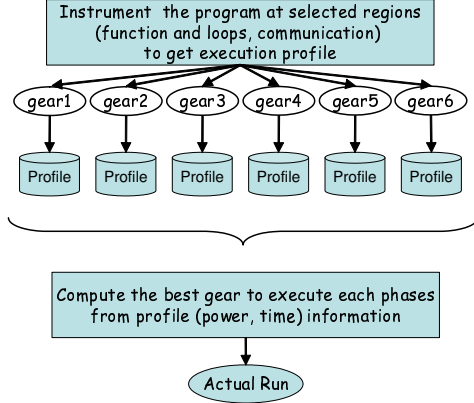


Figure 2. Flow of profile-based power-performance optimization

second). Brooks et. al [3] suggests that the metrics of future high-performance computing is not only performance, but also the power delay product (PDP) or energy delay product (EDP).

We use both metrics for our evaluation. PDP means the energy for executing an application, obtained by the summation of the product with time and actual power consumption. PDP is appropriate for the evaluation of low-power systems, such as a laptop or mobile device in which the battery life is the main concern for energy efficiency.

However, even if the energy is reduced, we should avoid a big performance loss in a high-performance system. Since our target is high-performance and low-power cluster computing, it is not enough to use only PDP, but also EDP. Our purpose is reducing the amount of power consumption and improving the power-performance efficiency in a high-performance computing system without performance loss. The EDP is a metric suitable for a high-end server class system. It is weighted by the execution time to take the performance into account in addition to the energy. EDP can be defined as follows:

$$EDP = T_{execute} \times Energy \quad (1)$$

In this metric, a lower EDP value means higher power-performance efficiency to run a program. In this paper, we schedule the clock frequency in order to reduce EDP.

3.2 The Power-performance Optimization Algorithm

We divide the program P into several regions, R_i . Each region is defined by instrumenting the program at appropriate locations. Then, we measure the execution time and

the power consumption for each region at each gear by our profiling tool in trial runs.

The optimization problem is to determine a set of clock frequencies, f_i , to minimize the estimated EDP $E(P)$ for the program P that is divided into n regions, R_i , $i = 1 \dots n$:

$$E(P) = \sum_{i=0}^n E(R_i, f_i) + \sum_{i=0}^n E_{trans}(R_i, f_i) \quad (2)$$

Here, $E(R_i, f_i)$ is the value of EDP when executing the region R_i at the clock frequency f_i . The value of f_i must be one of the clock frequencies given by each possible gear of the processor. $E_{trans}(R_i, f_i)$ represents the transition overhead in terms of the EDP to change a gear to the clock frequency f_i . This overhead depends on the clock frequencies of the adjacent regions of R_i . If the adjacent regions have the same clock frequency, this overhead becomes zero because the gear does not change. Actually, E_{trans} is the estimated EDP calculated by the product of the transition time and the power consumption during the transition. Note that, if we want to minimize the energy, that is, PDP, we can use the PDP value for $E(R_i, f_i)$ and $E_{trans}(R_i, f_i)$.

Our strategy to optimize $E(P)$ is to determine the clock frequency from the region having the largest EDP, because the region of the largest EDP is expected to give the largest contribution to the reduction of EDP in the total EDP. The algorithm decides the set of clock frequency F_i to give the minimum estimated EDP $E(P)$. Our algorithm optimizes the $E(P)$ in the following steps:

- Step 1: Initialize the clock frequency F_i for each region R_i as “not defined.” The value F_i indicates the clock frequency at which region R_i is to be executed.
- Step 2: Obtain $E(R_i, f_i)$ for each R_i and f_i of each gear from the execution profiles of the trial runs.
- Step 3: Choose the region R_i which has the largest EDP, $E(R_i, f_i)$, with F_i “not defined.” If all F_i are already defined, then stop.
- Step 4: Calculate the $E(P)$ for each gear of f_i when R_i is executed at the clock frequency f_i . For calculating $E(P)$, if the clock frequency of a region is not defined yet, the EDP value and the overhead for the region can be counted as zero. If the clock frequency of the adjacent regions to R_i is already defined and is different from the selected clock frequency for R_i , we must add the overhead for changing the gear to $E(P)$. Finally, choose the best f_i as F_i , which minimizes $E(P)$, and mark the region as “defined.” The clock frequency F_i is to be used to execute the region R_i .
- Repeat from Step 3 until all F_i are defined.

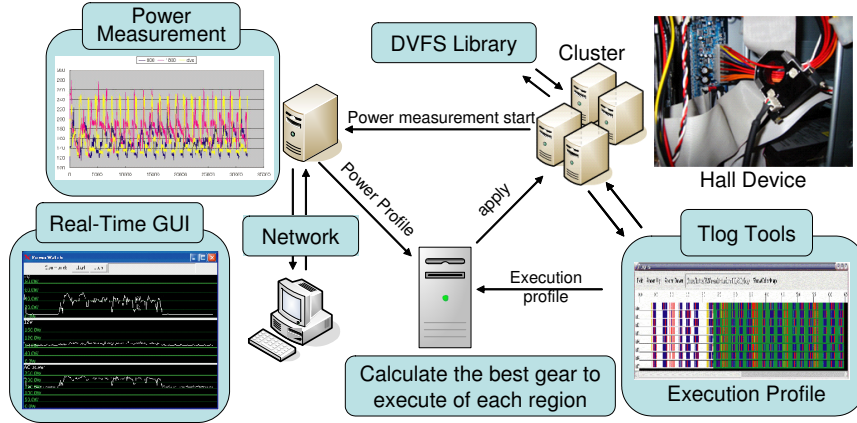


Figure 3. PowerWatch: Overview of the measurement system

Table 1. Voltage and frequency of Crusoe

gear	Clock	Voltage	FSB	TDP
1	933MHz	1.35V	133MHz	9W
2	800MHz	1.25V	133MHz	7W
3	667MHz	1.20V	133MHz	5W
4	533MHz	1.10V	133MHz	4W
5	300MHz	0.90V	100MHz	3W

Table 2. Voltage and frequency of Turion

gear	Clock	Voltage	TDP
1	800MHz	0.90V	9W
2	1000MHz	1.00V	–
3	1200MHz	1.05V	–
4	1400MHz	1.10V	–
5	1600MHz	1.15V	–
6	1800MHz	1.20	25W

3.3 Implementation

Figure 2 shows the flow of our power-performance optimization. It starts by instrumenting the program at an appropriate location to define the regions. The instrumented code is then executed, generating the profile of the power and execution time for each region. Once the profiling is finished running at each gear, we obtain $E(R_i, f_i)$. Then, we determine the best clock frequency F_i for each region by using the optimization algorithm described in the previous section. Finally, we apply the obtained set of clock frequencies in actual runs. The DVS system calls are called at the beginning of each region if the gear must be changed.

It is important to determine how and where to define the regions. When using our algorithm, we can insert profile code at arbitrary locations. For instance, we might insert profile code at each function or each loop, even at each statement. Our algorithm may remove unnecessary changes of gear by taking the overhead of the transition into account. We should, however, avoid too fine-grained instrumentation because it may cause large overhead of profiling and also perturbation of the execution time, resulting in inaccurate profile information. Currently, we insert the profile code manually. Our strategy of instrumentation is to focus on the communication in parallel programs. When a node waits for communication from other nodes, we can reduce the power dissipation by setting the clock frequency of the processor to be low. We define computation parts and communication parts as different regions. Another possible region may be memory-intensive regions, since the memory-intensive part may be executed at a lower clock frequency with less perfor-

mance loss. Automatic instrumentation would be desirable and will be investigated in future work.

3.4 PowerWatch: A Power Profile System

We designed and built the power profile system, PowerWatch, to measure and collect the power information of each node in a cluster. Figure 3 illustrates our system. The system includes the following three components:

- Power-monitoring system using the Hall device: This enables us to measure the electric current at several points in a few micro-seconds of resolution without any modification of the target hardware. The current system can monitor the electric current of 48 power-supply lines so that we can measure the power consumption of all 8 nodes at the same time. Several runtime APIs for an application program are provided to obtain the power profile of each node. A GUI power real-time monitoring system is also provided.
- Tlog execution profiling tool: “Tlog” stands for “timing log.” A set of library functions are provided to define events and generate event log records with a timestamp as the execution profile in an MPI parallel program. The regions in the program are defined by inserting the tlog function call at a specified location. The visualization tool “tlogview” is also provided to view the execution profile.
- DVFS control runtime library: we have instrumented several library functions to control DVFS.

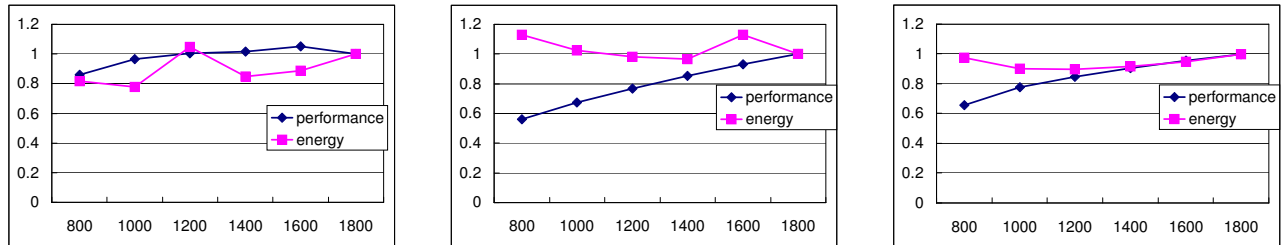


Figure 4. Characteristics of the Turion cluster on each gear with NPB IS (left),CG (center),FT (right)

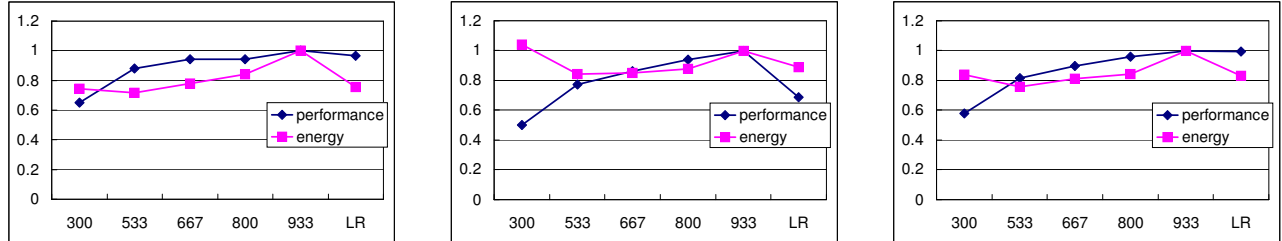


Figure 5. Characteristics of the Crusoe cluster on each gear with NPB IS (left),CG (center),FT (right)

We can obtain the execution time and power consumption of each region from the “tlog” execution profile and the power profile of a parallel application running in a cluster.

4 Experiment

4.1 Low-power Clusters

We examined the effectiveness of our algorithm on two types of power-scalable clusters (clusters using Crusoe and Turion). There are many processors which support DVFS, such as Pentium-M, ARM, and Intel’s XScale. We chose Turion because it supports dual-issue floating-point calculations, SSE2, SSE3, and 64-bit instructions that are useful for HPC applications. Crusoe is the first IA-32 compatible processor with a DVFS facility called LongRun. LongRun changes frequency and voltage automatically according to the CPU status.

The processor in the Turion cluster is Turion MT-32 (1.8 GHz with 64-KB L1 cache, 1-MB L2 cache, TDP is 24 W) with 1-GB DDR memory. The number of nodes is eight. These nodes are connected by Gigabit Ethernet (Dell PowerConnect 2442, Intel Gigabit Ethernet Card).

The processor in the Crusoe cluster is Crusoe TM-5800 (933-MHz with 64-KB L1 cache, 512-KB L2 cache, TDP is 9 W) with 256-MB SDR memory. The number of nodes is four. The network interface of the Crusoe cluster is a Fast Ethernet, a good balance between the processor and network. The Linux 2.6.11 kernel was installed on both clusters. All of the benchmarks were compiled with gcc version 3.4.11 with the LAM MPI 7.1.1 library. We measured the overhead of the DVFS function, and we have found it is nearly 50 μ seconds. The overhead of the DVFS function of Crusoe is processor defined. In the optimization of our algorithm, the overhead of the DVFS transition is set to 50 μ seconds in each processor according to our earlier measurement. Table 1 and Table 2 show the voltage and frequency

settings we used in Crusoe and Turion.¹

4.2 Benchmark and Power-performance Characteristics

We used NPB (NAS Parallel Benchmarks) version 3.1 for evaluation. We measured the execution time of FT, MG, CG. Communication has a big impact in reducing the energy or EDP. FT represents a memory-intensive application. CG represents a communication-intensive application with high frequency, peer-to-peer communication. IS is also a communication-intensive application.

First of all, we measured these benchmarks on the clusters at each gear to understand the power-performance characteristics of the clusters.

Turion cluster: Figure 4 shows energy and performance normalized to the standard frequency at each gear. At the low gear, we can reduce the energy in some benchmarks. The performance of IS and FT does not have a large difference between different gears while CG goes down by a lower gear.

Crusoe cluster: Figure 5 shows the energy and performance characteristics of each benchmark in various gears. We found the ratio of energy reduction, and it is lower than in LongRun which indicated by “LR”. At a gear of 300 MHz, the performance of all benchmarks rapidly goes down. The reason is not only the declining clock frequency, but also the FSB clock frequency goes down from 133 MHz to 100 MHz. The performance of CG is not good using LongRun.

4.3 Our Strategy of Defining Regions

The most important strategy of defining regions is to focus on the communication region. In parallel applications, communication takes a large amount of execution

¹AMD does not announce some details of Turion’s frequency’s voltage, so we assumed the settings of these voltages.

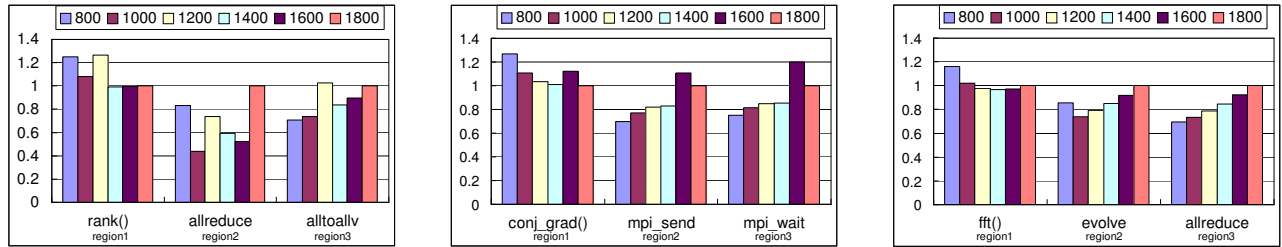


Figure 6. Details of the energy on each gear in the Turion cluster with NPB IS (left),CG (center),FT (right)

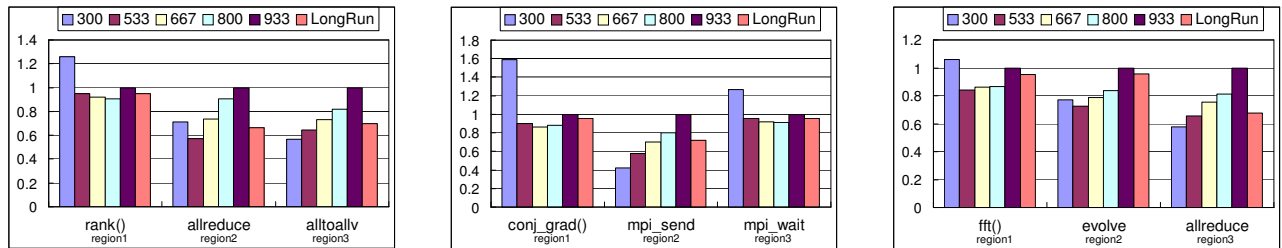


Figure 7. Detail of the energy on the Crusoe cluster in each clock with NPB IS (left),CG (center),FT (right)

time; therefore, by using a lower gear, wasted energy can be greatly reduced. So, we instrument the communication region (such as MPI_Alltoall), and we especially focus on Alltoall communication. We define most of the communication function in the main iteration as a region. By this strategy we define the region of the program as follows: MPI_Allreduce, MPI_Alltoall, MPI_Alltoallv in IS, MPI_Allreduce in FT, MPI_Irecv, MPI_Send, MPI_Wait in CG and MG. Particularly, IS and FT includes Alltoall communication, which spends a large amount of time in general, so that by using a lower gear we expect to reduce energy without performance impact. CG and MG includes MPI_Wait to wait in the idle state, so that we can reduce energy with a lower gear.

Other strategies also focus on the memory access region and large computation region. We assume that a processor does not need a higher gear in the case of off-chip memory access. By using a lower gear when a processor needs off-chip memory access, we can reduce energy with a small performance impact. With this strategy we also define the regions of the programs as follows: rank() in IS, fft(), evolve(), checksum() in FT, conj_grad() in CG, mg3P(), resid() in MG. Particularly, fft() generally needs off-chip memory access frequently. Therefore, by using a lower gear, we expect that we can reduce more energy.

Turion cluster: Figure 6 shows energy of each region normalized to the standard highest gear. We only illustrate the regions which execute for a long time. The energy of IS and FT becomes smaller in some region at a lower gear. The communication region can reduce energy by using a lower gear. The fft() function does not increase energy very much by using a lower gear.

Crusoe cluster: Figure 7 shows energy of each region

normalized to the standard highest gear. In the case of communication, a lower frequency can reduce energy dramatically (over 40%) However, LongRun can adjust to a better gear to reduce the energy.

4.4 Results of Optimization

Turion cluster: Table 3 shows the set of clock frequencies of each major region estimated by our algorithm, as indicated with Figure 6 (from the left, MG: resid() as region1, mg3P() as region2, mpi_send as region3). We optimized for two values, energy-aware optimization (This means E(P) is energy), and EDP-aware optimization (This means E(P) is EDP). In the case of energy-aware optimization, we found that some regions, particularly most communication regions, select a lower gear. In the case of EDP-aware optimization, CG and MG does not need to change the clock frequency.

Figure 8 shows the estimated result and measured result of applying our optimization normalized to estimated result (left: Energy-aware, Right: EDP-aware). Our estimated result is almost the same as the measured result. In the case of the energy-aware optimization, the difference between the estimated and measured result is small. However, CG's execution time becomes large. We presume this is a perturbation caused by the DVFS overhead. In the case of EDP-aware optimization, the difference between the estimated and measured result is small (CG and MG does not change gears, so these are the same).

Figure 10 shows the optimization result of PDP, EDP and the performance normalized with the standard highest frequency. All benchmarks can reduce the energy and EDP using our method; in particular, IS and FT are about 20%. While energy-aware optimization may cause large performance loss, the performance impact of EDP-aware opti-

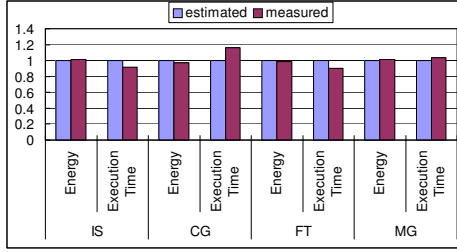


Figure 8. The difference between the estimated and measured result on the Turion cluster

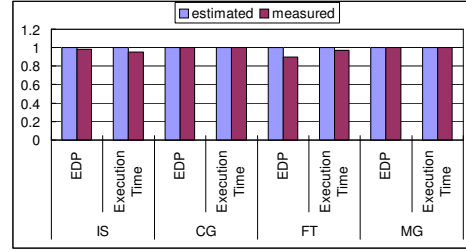


Figure 9. The difference between the estimated and measured result on the Crusoe cluster

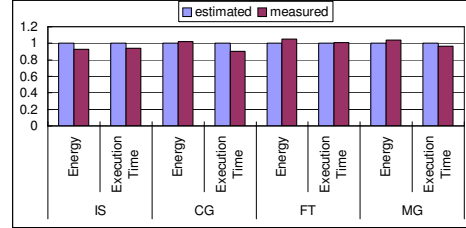
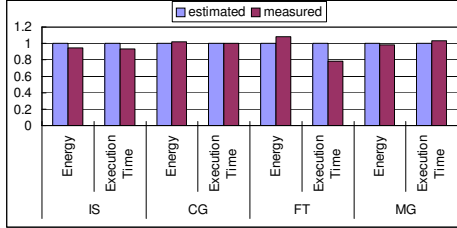


Table 3. Selected Frequency of Turion

		region1	region2	region3
IS	PDP	1800	1000	800
	EDP	1800	1000	1000
FT	PDP	1400	1400	1000
	EDP	1800	1800	1000
CG	PDP	1000	1600	1400
	EDP	1800	1800	1800
MG	PDP	1400	1400	1400
	EDP	1800	1800	1800

Table 4. Selected Frequency of Crusoe

		region1	region2	region3
IS	PDP	933	533	300
	EDP	933	533	300
FT	PDP	533	800	533
	EDP	800	533	300
CG	PDP	533	533	800
	EDP	800	800	800
MG	PDP	667	533	533
	EDP	800	800	800

mization is not very large (within a few percentage points of the decline with EDP-aware optimization). EDP-aware optimization also reduces both energy and EDP effectively.

Crusoe cluster: Table 4 shows the estimated set of clock frequencies of major regions indicated by Figure 7. In the case of energy-aware optimization, some regions use a lower gear, as in the Turion cluster. In the case of EDP-aware optimization, CG and MG do not need to change the clock frequency in major regions and in IS, we obtain the same results as in the energy-aware optimization result.

Figure 9 shows the estimated result and measured result of applying our optimization normalized to estimated result (energy-aware and EDP-aware). In the case of energy-aware optimization, our estimated result is almost the same as the measured result, but the execution time of CG is dif-

ferent from the measured result. In the case of EDP-aware optimization, our estimated result is almost the same without the FT execution time. But, EDP is not much different from the estimate.

Figure 11 shows the optimization result of both optimizations normalized with a standard frequency. We can obtain a better benefit in the Crusoe cluster rather than in the Turion cluster. That is because the Crusoe cluster uses Fast Ethernet. It takes the ratio of the communication region to be large in the execution of a program. Another reason is the ratio of power consumption of the processor in the system. While the CPU power consumption of the Turion cluster consumes about 60%, the Crusoe cluster consumes nearly 90%. As a result, in all the benchmarks, we can reduce the energy by about 20% (IS is about 40%) The energy-aware optimization incurs performance loss. In the case of CG, when using LongRun, EDP rapidly increases with EDP-aware optimization. This indicates that the performance of the CG benchmark is not good with LongRun.

5 Discussion

We applied our algorithm to two types of values. Although energy-aware optimization has the benefits of energy reduction, the performance impact may become large; therefore, it may not be suitable for HPC power-aware computing. On the other hand, EDP-aware optimization has the benefit of EDP reduction without much performance impact (only a few percentage points). In this result, our algorithm can almost estimate the actual measured result. In most of the cases, our algorithm estimates a lower value than that of an actual run.

The transition cost of DVFS is not small. Before our evaluation, we measured DVFS overhead to measure the time. Sometimes we found a large slack time of about 300 μ seconds while almost spending about 30 μ seconds. Thus, in this experiment, we set the transition time to 50 μ seconds (the minimum interval to set MSR).

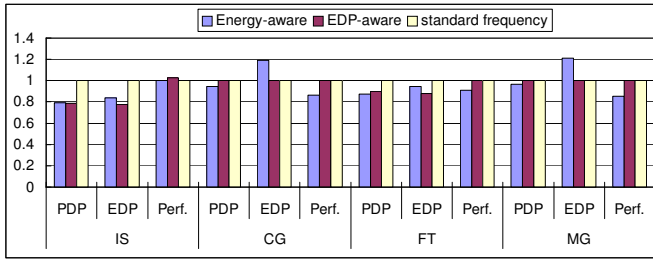


Figure 10. PDP, EDP and the performance of our method normalized with a standard frequency on the Turion cluster

In terms of power consumption, Turion's power is quite lower than that of Opteron or Athlon. Opteron and Athlon also have DVFS called PowerNow!, but the voltage is more than 0.3 V higher than that of Turion. For example, TDP of the Opteron146 processor consumes 89W, while at the lowest gear it consumes only 28 W (Turion MT-32 consumes 24 W maximum, and 7.7 W minimum) [2]. We believe in the case of Opteron (and some high-end processors), we obtain a dramatic effect of energy and EDP reduction by using our method.

And, the network interface has a big impact in power-aware computing. In this paper, the network switch's power consumption and energy were not included in our evaluation. The reason is that a network switch constantly dissipates power consumption. And the number of nodes of our cluster is only 8, but a network switch has 24 ports. Also, network power would make evaluation unfair in the case of an 8-node cluster. This is a trade-off between performance and energy and cost.

Finally, we are interested in optimizing and estimating power in a large-scale system. Recently, HPC systems have become large with thousands of processors. It is difficult to measure the energy of each node. Currently, we are working on estimating the power-performance characteristics of a large-scale cluster from the result of a small cluster. This work will help us to optimize the energy dissipation of a large-scale system.

6. Concluding Remarks and Future Work

In this paper we proposed a profile-based power-performance optimization by using DVFS in an HPC cluster. We divided the program execution into several regions and determined the best gear to execute each region according to the power and execution profile information from a trial run. For our evaluation, we designed and built the environment, PowerWatch.

Our proposed algorithm of selecting the best gear in each region takes the overhead of the DVFS transition into account to achieve high power performance. We examined two types of power-scalable clusters. Our algorithm increased the power performance in both clusters, especially in the case of the Crusoe cluster (about 40%).

In this study, we instrumented the code manually. As our

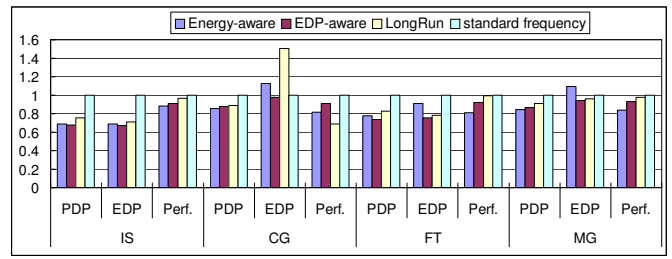


Figure 11. PDP, EDP and the performance of our method normalized with a standard frequency on the Crusoe cluster

future work, we will develop a system which instruments the code to the program automatically. We instrumented the program at the function level. It could successfully reduce energy and EDP. However, we can reduce the energy and EDP more effectively to analyze the application behavior (memory access, cache miss, etc.). Next, we will evaluate other applications, such as a server system benchmark or database application, which really needs low energy dissipation for managing cost or fault recovery.

References

- [1] N. Adiga, G. Almasi, G. Almasi, Y. Aridor, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, M. Blumrich, A. Bright et.al. An overview of the bluegene/l supercomputer. In *Supercomputing Conference 05 (SC'05)*, November 2005.
- [2] AMD. Corp. AMD athlontm 64 processor power and thermal data sheet, 2005.
- [3] D. Brooks, P. Bose, S. Schuster, H. Jacobson, P. Kudva, and P. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. In *IEEE Micro*, volume 20, pages 26–44, 2000.
- [4] Transmeta Corp. Longrun thermal management, 2001. <http://www.transmeta.com/>
- [5] R. Ge, X. Feng, and K. W. Cameron. Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. In *Supercomputing Conference 05*, November 2005.
- [6] H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka, D. Takahashi, Y. Hotta. Megaprote: 1 TFlops/10kw rack is feasible even with only commodity technology. In *SC 2005 Conference (SC'05)*, November 2005.
- [7] Y. Hotta, M. Sato, T. Boku, D. Takahashi, Y. Nakajima, and C. Takahashi. Measurement and characterization of power consumption of microprocessors for power-aware computing. In *CoolChips7*, April 2004.
- [8] C. Hsing Hsu and W. Chun Feng. A power-aware run-time system for high-performance computing. In *Supercomputing Conference 05*, November 2005.
- [9] N. Kappiah, V. W. Freeh, and D. K. Lowenthal. Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs. In *Supercomputing Conference 05*, November 2005.
- [10] M. Warren, E. Weigle, and W. Chun Feng. High-density computing: A 240-processor beowulf in one cubic meter. In *Supercomputing Conference 02*, November 2002.