A Node Level Performance/Power Efficiency Aware Resource Management Technique

LE LI[†1] Keiichiro FUKAZAWA[†2] Hiroshi NAKASHIMA[†2]
Takeshi NANRI[†3]

Abstract: The amount of data grows rapidly, since everything can be easily digitalized today. For obtaining values from those data, datacenters continuously enlarge and enhance their computer systems. However, recently the silicon transistors are difficult to develop, and it becomes much harder for us to extract its physical ultimate, and so the performance becomes hardly catch up the growth of data. Thus, we have to increase the number of hardware to reach the demanded performance and also sacrifice more power as a result. Most of the supercomputer systems are homogeneous systems, which means all of their nodes are comprised by identical hardware. However, the process variation in manufacturing processes results in a variation on performance/power. In this research, taking a node level performance power variation into consideration, we come up with a power saving resource management technique. We also make a job scheduling simulator to verify its effects by comparing with primary methods.

Keywords: Power Saving, Resource Management, Node Level Heterogeneity, Job Scheduling

1. Introduction

Increasing energy consumption has become a major challenge for the development of HPC systems. This has been aware by not only data centers and system makers, but also governments and environmental organizations since high power consumption also means high cost and environmental matters.

It is estimated that the power consumption of data centers will reach 2% of the world's total electricity usage by 2020 [1]. For controlling a power budget, data centers have to limit their system's specification or carry out policies that impose restriction on peak power, and thus performance is also limited at the same time. However, more powerful systems are exactly demanded for driving future computing, especially in some fields, such as big data and machine learning.

For competing in a data-driven world, a great amount of systems were developed and updated. From 1993 to 2018, the sum performance of all the recorded HPC systems in Top 500 increased a thousand times, with a 20-fold average increase every 5 years [2]. Additionally, the performance of the top 10 systems averagely rises 90% every year, meanwhile, power consumption also rises 20% every year in average [3].

The purpose of this paper is to analyze the power consumption situation and tendency of HPC systems, come up with a resource management technique that improves power efficiency, without leading increase of execution time. The remainder of this paper is organized as follows. Section 2 illustrates HPC systems' power situation in the past several years and the latest predictive inclination. Section 3 describes the node level heterogeneity and its genetic analysis in large-scale systems and the evaluation of it, which have been utilized to improve power efficiency by our creation. Section 4 elaborates the organization of node level heterogeneity aware resource management heuristic. Section 5 introduces the simulator we have created and the evaluation simulation. Section 6 makes a conclusion for the contents in the paper.

†1 Graduate School of Informatics, Kyoto University
†2 Academic Center for Computing and Media Studies, Kyoto University
†3 Research Institute for Information Technology Kyushu University

2. Background

2.1 Performance/Power Tradeoff

Fig. 1 shows the historical data of the top performance supercomputers. Even the performance and power efficiency benefit most from the development of hardware, an unavoidable fact is a tradeoff between performance and power. Note that for activating next generation supercomputer — the exascale system, about 50MW is required that judging from the prediction, which is almost 2.5 times of the target power (20MW) that planned by many system centers and organizations.

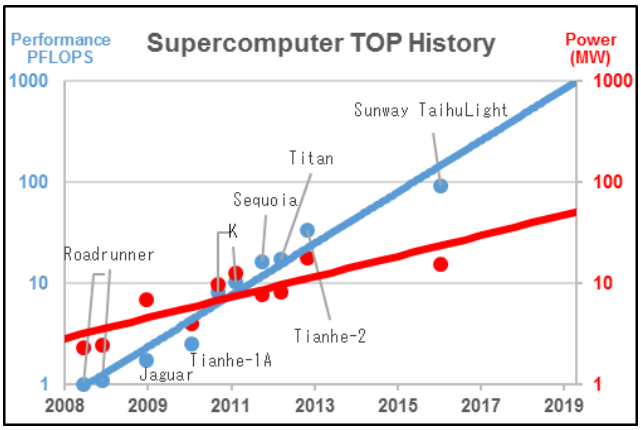

Fig. 1 LINPACK Performance & Power Data of Historical Top Performance Supercomputers [4]

2.2 Power Efficiency Enhancement

For achieving probable power saving and limiting the peak power , many researches have been studied. To cite recent examples, forcing processors to execute programs on optimal frequency to reach high power efficiency, i.e. DVFS and power capping, or dynamically tuning the resource configuration of

running jobs on power constrained systems [5]. Furthermore, other researchers try to do a prediction for the power consumption through job history before starting jobs [6] or predict system level power to avoid power surpasses threshold [7]. However, due to the internal overhead of power capping and prediction, most of these techniques may lead to time degradation.

3. Node Level Heterogeneity

Many powerful systems appeared in the past several years, and more gigantic systems will spring forth in the near future, many of them are large-scale systems that comprised by over thousands of nodes. A node is a basic unit in system network, which is comprised by fundamental hardware, like motherboards, processors and memories. Nodes in a specific system usually constructed under the same specification, which means they are comprised by hardware that has identical model and launch the same operating system. This is the so-called "Homogeneous System". Even though some researchers have started to undertake researches on the heterogeneous system architecture, like considering processing element and ACPI performance variability in Eurora system, whose nodes are built using heterogeneous hardware [8]. However, the homogeneous system still has a forceful momentum, and thus for the foreseeable future, it will remain mainstream. The most important reason is that the heterogeneity system is sophisticated, and thus hard to be built, and the homogeneous system has many advantages, like easy workloads balancing and program portability. The reasons that accounts for these are similar node performance, identical instruction set and software (OS, compliers).

Even though those systems are homogeneous systems, variation exactly exists. In this paper, we mainly discuss the node level performance/power variation.

3.1 Term Explanation

Node level Heterogeneity implies that even though nodes in a specific system are comprised by identical hardware model, viewing performance/power variation exists among nodes. In other words, when executing the same workload, different nodes may spend different time and have different power consumption. The gigantism of HPC systems results in the increase of physical nodes, and therefore, leads to the growth of node level variation. Some of those large-scale systems contain more than thousands of nodes, and thus have millions of processor cores and petabyte scale memories.

Although the power distribution varies from system to system and depends on the executing workloads. However, on the other hand, when executing workloads, Processing Elements (PE, the sum of memory and processor package, which refers to a small silicon block that fabricates microprocessor circuit) consume most of the power [9]. Processors need substantial power in driving its processing, and with the demand of operating large amount of data, memory capacity also has been expanded, especially in HPC systems. In some cases, it may reach hundreds of watt, and become a non-negligible existence.

3.2 Genetic Analysis

- ***Physical Reason***

As previously mentioned, with the gigantism of HPC systems, the node level heterogeneity become much more dramatic than ever before. There are several causes of node performance and power differences in spite of the same specification among nodes, one reason is the process variation of hardware in manufacturing processes.

For describing this, we should explore the manufacturing processes of processors. As it is well known to all that transistors are the most important part of a processor. Hundreds of procedures are needed for making sands into a silicon wafer, like polishing, etching, ion doping etc. [10]. For controlling error, all these steps are operated by precision instruments. However, transistors are extremely tiny (from microscale in the past to nanoscale now), to some degree, a small deviation will also show impacts if we consider it from the respect of percentage [11]. For example, the width and length will be different due to diffraction [12]. Worse still, because of the difficulties in scaling, process variation has exactly become a trouble in integrated circuit design. However, macroscopically speaking, process variation finally does impact on power efficiency in some degree, which is viewing.

- ***Technical Reason***

Besides the physical reason in making hardware, recent micro-techniques also result in runtime variation, comparing with "the external variation" among nodes, which can be named "the within-node internal variation". Those micro-techniques mainly refer to Dynamic Voltage Frequency Scaling (DVFS) and Turbo Boost etc.

DVFS is a widely used power management technique in integrated circuit design, where voltage and frequency is tuned depending upon runtime circumstances for reaching the balance between performance and power. Its function has been evaluated by many researchers [1] [13], and also utilized to improve power efficiency by other researchers, who have developed a DVFS-enabled scheduling system for a cloud data center [1].

Overclocking, which is also a dynamic frequency tuning technique that adjusts frequency in pace with the task changing, has been utilized to trade power consumption for performance in current processors design. In general, for a given processor, higher frequency implies higher performance, this technology can accelerate processors for current workloads, and turn into a low frequency status (which also means low power consumption) to save power. Its functions also have been studied on the respect of HPC systems [14]. Unfortunately, the relation between performance and power is non-linear [15], thus when processors are accelerated to operate workloads in a high frequency, it may result in a bad power efficiency.

3.3 Statistical Verification

The process variation in making hardware and technical reason leading runtime variation have been illustrated in previous contents. A more important point is how much they can influence performance and power.

A significant element for scaling heterogeneity, is the

number of nodes. Hence, we evaluated the node level variation on two many-nodes systems with two different measuring tools. The first one is the system-A of Kyoto University (Camphor), with Cray-Pat, and the second one is the subsystem-A of Kyushu University (ITO), with Intel RAPL.

- ***Benchmarks***

Two benchmarks were used for scaling the performance and power variation on processors and memories. One is High Performance Conjugate Gradients (HPCG) [16], the other one is STREAM [17].

HPCG is designed as a complement to High Performance LINPACK (HPL) to create a new metric for ranking HPC systems and was exploited by Top 500 Supercomputer Site from November 2017. It mainly carries out sparse matrix-vector multiplication, vector updates etc., which is the so-called computational intensive program. STREAM measures sustainable memory bandwidth, and carries out simple vector kernel computation. Different from HPCG, STREAM is a memory intensive program.

In the System-A of Kyoto University, we simply used them to evaluate the node level heterogeneity, and in Subsystem-A of Kyushu University, we also evaluate whether the type of benchmarks and their problem sizes influence the variation.

- ***System-A of Kyoto University (Camphor)***

Table 1 shows the specification of this system. Each node has only one socket that incorporated with one latest Intel Xeon Phi series coprocessor, code name Knights Landing (KNL). **Table 2** shows the benchmarking information. We executed those benchmarks in single node mode, and measured execution time and power consumption by Cray-Pat performance tool kit. Due to the job scheduling policy of the system, only 414 samples for HPCG, and 341 samples for STREAM were measured even though there are 1800 nodes.

Fig. 2 shows the measurement results of HPCG. The x-axis stands for samples, which refers to each measured node. The left y-axis is the power variation, and the right y-axis is the time degradation, both are the comparison to the best case (i.e. the least power or time consuming node). Power stands for Processing Element Power. Both power consumption and time expenditure are varying among nodes. When executing HPCG, the power variation reaches 15% and the time degradation of most of the nodes is under 8%.

Fig. 3 shows the measurement results of STREAM. Similar with HPCG, the power variation of STREAM also reaches a high level (almost 12%) varying among nodes. However, the time degradation of most of the nodes is around 3%, which implies that the execution time of most of the nodes is closed.

- ***Subsystem-A of Kyushu University (ITO)***

Table 3 shows the specification of this system. Different with the system-A of Kyoto University, the node of subsystem-A of Kyushu University has two sockets, and each socket incorporates one Intel Xeon Gold 6154 processor. The evaluation is also carried out by utilizing HPCG and STREAM. In addition, different problem sizes in HPCG and STREAM also be applied to see its impact on variation. **Table 4** shows the benchmark testing information. In the system-A of Kyoto

Table 1 Specification of Kyoto University System-A

The Number of Nodes	1800
Processor	Intel Xeon Phi Knights Landing (68 cores 1.4GHz ×1) / node
MCDRAM	Cache Mode
Memory	DDR4-2133MHz 96GB / node
Measurement Tool	Cray-Pat

Table 2 Benchmark Testing Information

Benchmark	**Problem Size & Parallelism**
HPCG	Problem Size: Default 64 processes (1 core/1 thread for each)
STREAM	Problem Size: Default 68 processes (1 core/1 thread for each)

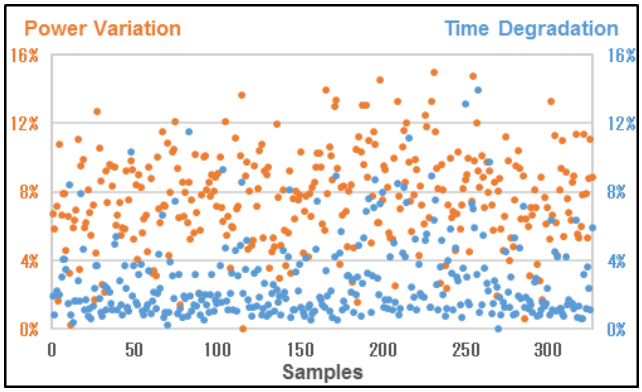

Fig. 2 Measured Power/Time Variation of HPCG

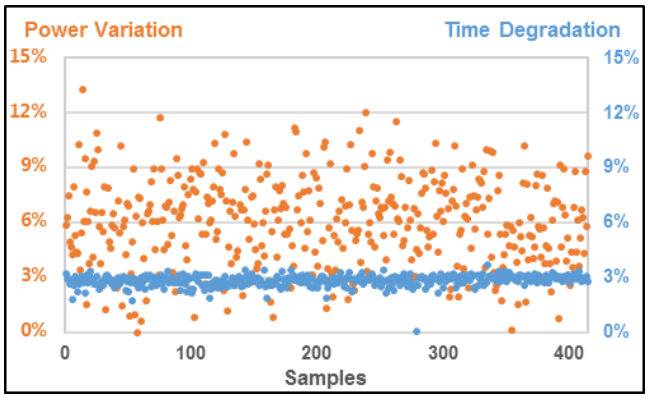

Fig. 3 Measured Power/Time Variation of STREAM

Table 3 Specification of Kyushu University Subsystem-A

The Number of Nodes	2000
Processor	Intel Xeon Gold 6154 (Skylake-SP) (3.0 GHz, 18 core) ×2 / node
Memory	DDR4-2666 192 GB / node
Measurement Tool	Intel RAPL

Table 4 Benchmark Testing Information

Benchmark	**Problem Size & Parallelism**
HPCG	Problem Size 1: X=104, Y=104, Z=104 1 processes (36 cores/36 threads for each)
	Problem Size 2: X=128, Y=192, Z=128 1 processes (36 cores/36 threads for each)
STREAM	Problem Size 1: 86GB 1 processes (36 cores/36 threads for each)
	Problem Size 2: 144GB 1 processes (36 cores/36 threads for each)

University, we only measured a small part of nodes. In this system, we measured all of them. Another change in this testing is that PE power was divided into package power and memory power for scaling respective power variation.

- ***Power***

Fig. 4 shows the measured power of each workload. The x-axis refers to nodes, which is sorted by average power (average value of all the four workloads) in ascending order. The y-axis stands for package or memory power.

The data of package power illustrates that for STREAM benchmark, those nodes consume less power when executing problem size 1, most of them also consume less power when executing problem size 2. However, drawing from the measurement result of HPCG benchmark, it becomes different, those nodes consume less power when executing problem size 1, they do not always consume less power when executing problem size 2.

Comparing the data of package power, the memory power is more stable, in other words, those nodes consume less power when executing one workload, they also consume less power when executing other workloads.

- ***Energy***

Power only stands for instantaneous expenditure, a more important quota which we concerned is, considering execution time as a factor, how much energy a node consumes when executing a specific workload.

Fig. 5 shows the measurement of package and memory energy consumption. The x-axis stands for nodes (ordered by average energy consumption), and the y-axis stands for variation, which is shown in the percentage of the energy consumption of each node comparing with average energy consumption of all the nodes.

In the previous package power variation figure, when a node executes different workloads, it may show different inclinations, which can be named within-node variation. On the other hand, we can find the inclination (i.e. those nodes consume less energy when executing one workload, most of them also consume less energy when executing other workloads) when looking into the total energy consumption within the execution period. Thus it is memory energy variation.

Most of the data are around the average value of energy consumption. For example, the orange data (HPCG – 128 192 128(XYZ)) is within the range of its average ±10%.

A more interesting phenomenon is that the variation level of package energy consumption depends on the benchmark and its problem sizes. For instance, the orange data (HPCG – 128 192 128(XYZ)) shows that the maximum variation is almost 18% (from 90% to 108%) and other benchmarking data only shows 10% variation (from 93% to 103%). Moreover, these data may illustrate that the package energy variation of computational intensive workloads is greater than that of memory intensive workloads.

Even though it is obvious that the variation level of package energy consumption is depended on the workloads, however, the memory energy variation is different. To some degree, all the benchmarking data shows almost 40% variation (from 80% to

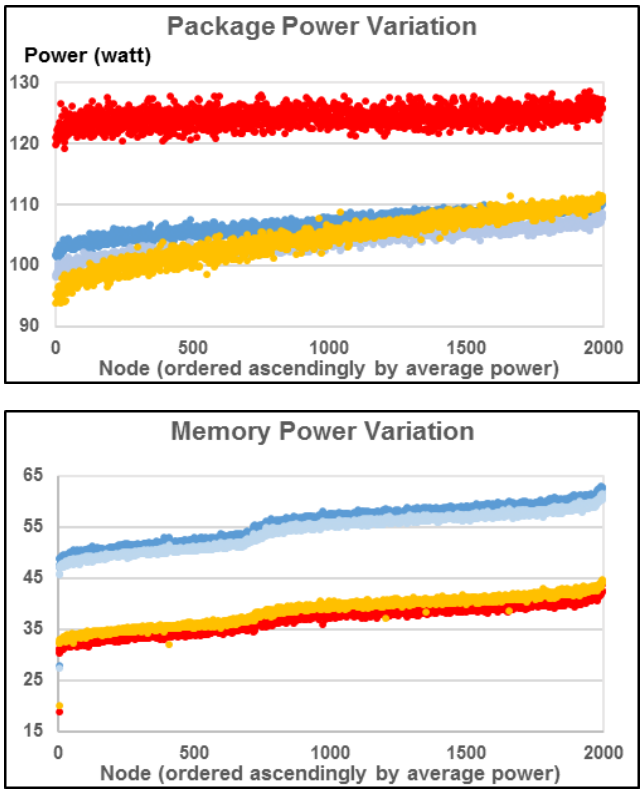

Fig. 4 Package and Memory Power Variation

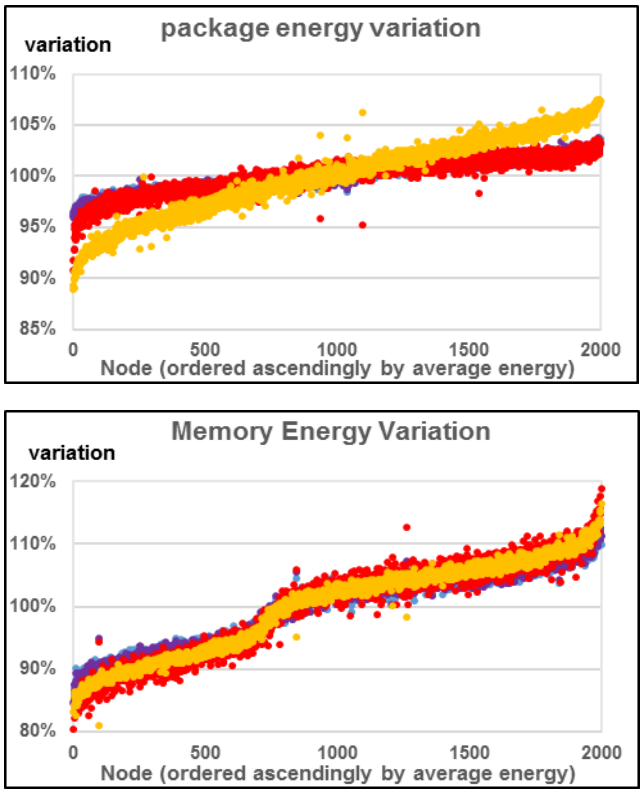

Fig. 5 Package and Memory Energy Variation

120%).

With this knowledge, we can consider a node level heterogeneity aware scheduling policy, for example, the most straightforward method is scheduling those low power consuming nodes precedentially, to achieve probable energy saving.

4. Technique

Resource management technique and HPC scheduler have a long history and have been well developed for HPC systems, which are growing much more gigantic than ever before. Those techniques are important for increasing systematic efficiency.

In this chapter, we design a node heterogeneity aware resource management heuristic and compare its function with two different heuristics. The first comparison target decides execution nodes randomly (like the scheduler in the systems of Kyoto University), and the second one selects execution nodes by node-id (like the Kyushu University systems).

4.1 Hierarchy Pool

Based on our introduction in chapter 3, if we increase the usage probability of low power consuming nodes and avoid passing jobs to those high power consuming nodes, we can achieve probable power saving in some degree.

Differ from other node pool management methods, we manage nodes in hierarchy pools. This heuristic is based on the testing result which has been shown in chapter 3, and the pool arrangement is shown in **Fig. 6**.

In this heuristic, three pools were prepared, and arrange nodes by their power efficiency (P_e). From upper to lower is High Efficiency Node Pool (*Pool 1*), General Node Pool (*Pool 2*), and Low Efficiency Node Pool (*Pool 3*). It can be expressed, as:

$$P_e(Pool1) > P_e(Pool2) > P_e(Pool3)$$

When the scheduler intends to start a new job, *Pool 1* will be firstly checked. If those nodes in *Pool 1* are all unavailable, or the number of nodes is not enough, those nodes in *Pool 2* will be chosen. This policy is also applied between *Pool 2* and *Pool 3*. When nodes are chosen, they will be put into a corresponding pool in Exclusive Pool, and when it finishes execution, it will return to its primary pool in Free Pool. Inside each pool, nodes are chosen randomly. With this resource management, those nodes which are high power efficiency node will be exploited before others that are low power efficiency node.

4.2 Pool Size

Another problem should be considered is how to decide the pools' size. The target of many HPC system centers is to reinforce the function of systems as much as possible, systems are launched almost the whole years, and whenever it will reach a minimum usage. However, most of the time, systems are not fully loaded (i.e. 100% usage).

Hence, over a period of time, the usage of one system is between its lowest usage (U_L) and highest usage (U_H). We can set the size of pools, as:

$$Size(Pool1) = U_L$$
$$Size(Pool2) = U_H - U_L$$
$$Size(Pool3) = 100\% - U_H$$

This is a clear thought that all the nodes in *Pool 1* will be in execution status when the system is active since within this period, the usage of system is greater than U_L. And for those nodes in *Pool 3*, they are avoided to be selected for completing workloads, which is because those nodes in *Pool 2* will not be exhausted when usage is under its peak.

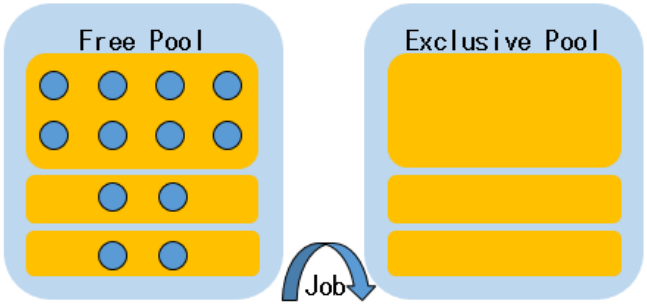

Fig. 6 Hierarchy Node Pool Management

5. Simulation & Result

In this chapter, we will introduce our simulation and how much energy can be saved through applying our resource management technique. We utilized the system-A of Kyushu University (ITO) to carry out this simulation. However, it is impossible for a user to revise the real scheduler of a supercomputer, hence, we made a multi-node scheduling simulator and used it to evaluate our proposition.

5.1 Simulator

There already exist many simulators in the world and did contribution to science for many researchers. While most of those simulators have two factors that may lead to impact on our result. The first reason that accounts for this is they not only simulate a scheduler and its interior elements, but also simulate the hardware (like the processor, cache, memory etc.), and the second one need to be guaranteed is more important that many of them only schedule workloads, without actually executing those workloads or just execute workloads on the simulated hardware (the execution result is modeled). Our research focus on the hardware and its process variation, for reflecting them, the workloads should be actually executed on real nodes, and that is what our creation can do.

- ***Structure***

This simulator is created by utilizing the widely used parallel computing architectures MPI and OpenMP. There are mainly three components in this simulator, Resource Management Kernel (RMK), Workload Simulator (WLS), and Compute Node Kernel (CNK). The structure of it is shown in **Fig. 7**.

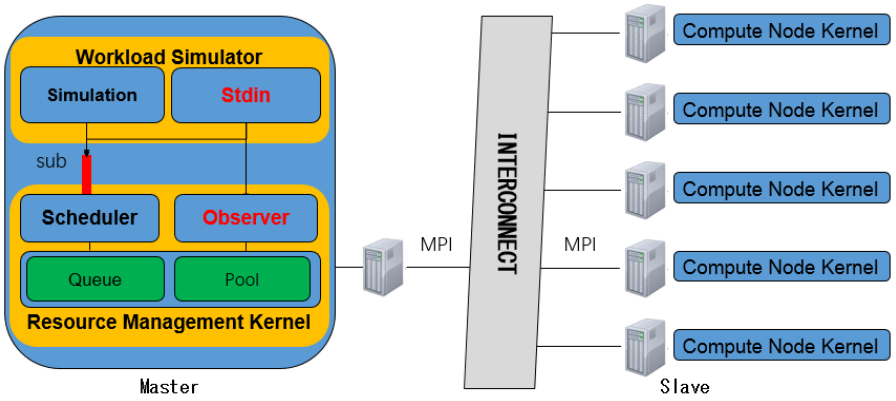

Fig. 7 Simulator Structure

- ***Resource Management Kernel (RMK)***

The RMK mainly contains Scheduler, Job Queue, and Node Pool. The Scheduler handle the submissions from WLS and push them into corresponding Job Queue if they are accepted. When there are waiting jobs in Job Queue, the Scheduler will select execution nodes from Node Pool, and then pass jobs through MPI communication, certainly, the precondition is that there are available nodes.

- ***Workload Simulator (WLS)***

The WLS is applied to generate workloads. When the WLS is launched, it reads workloads information from setup file, and other things that needed to be prepared are only executables.

- ***Compute Node Kernel (CNK)***

The CNK is very simple, it only receives workloads from RMK's scheduler and completes it. After finishing the execution, a signal will be sent to RMK via MPI communication.

- ***Execution***

When the simulator is launched, only one node will become the Master, the others will be treated as Slaves. The Master Node is decided by the setup file, judging from MPI rank or hostname. RMK and WLS will be launched on the Master node, the others launch CNK.

5.2 Workload

In chapter 3, we obtained the node benchmarking data by utilized HPCG and STREAM. In our simulation, we used 2D-Fast Fourier Transformation (FFT2D) [18] to carry out evaluation. For evaluating our creation in a real case, the node usage is an important element. Thus, if the usage is always very low or high, it cannot reflect the effect. Furthermore, if the usage is always 100% (like someone attempt to exhaust all of the nodes), scheduling will not produce many effects.

For simulating a real case, we traced node usage of the System-A of Kyoto University, and its data is shown in **Fig. 8**. This system carries out job exclusive mode (i.e. one node can only execute one job), generally, its usage is higher than those systems that carry out job busy mode (i.e. one node may execute multiple jobs at the same time). Drawing from the usage tracing figure, within the period that shows in the x-axis, the node usage ranges from 65% ~ 80%. The workloads information is shown in **Table 5**.

Table 5 Workload Information

Workloads	Execution Times
FFT2D (problem size 3600): 36 processes (1 core/1 thread for each)	6102
FFT2D (problem size 7200): 36 processes (1 core/1 thread for each)	25050

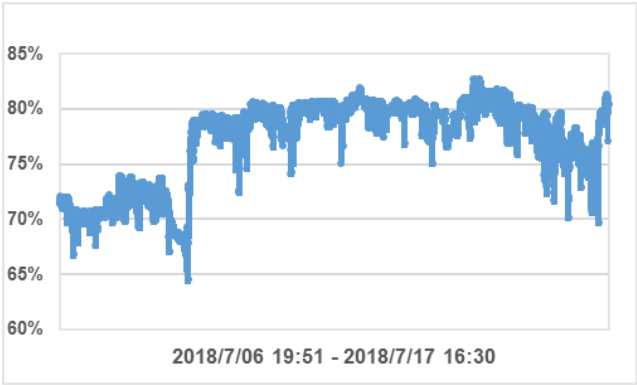

Fig. 8 Node Usage of The System-A of Kyoto University

5.3 Result

Table 6 shows the comparison of our creation with other node select heuristics. Random is the so-called heuristic that decides execution nodes randomly, and Sequence selects nodes by their node-id order. Within the period of simulation, the maximum usage of our simulation reaches 67%.

Our result clearly shows that whether compare with Random or Sequence, the Hierarchy resource management heuristic enhances not only package power efficiency, but also memory power efficiency. Within the period of simulation, this heuristic can achieve 0.640% energy saving when comparing with Random, and 1.346% energy saving when comparing with Sequence. These effects and workloads are highly reasonable dependent in two respects. The first reason that accounts for this is the node usage. The node level heterogeneity scale of available nodes is greatly based on the number of nodes in free pool, higher node usage implies less available nodes, which also means low heterogeneity scale. Another reason is the type of workloads. This results from the reason that we have described

in chapter 3, the heterogeneity scale not only depends on the hardware property of nodes, but also has relation with workloads.

Although, in this simulation, this technique only achieves 1% level saving now, it shows contribution in two aspects. Firstly, it reduces power consumption, without causing time degradation (the time variation is less than 0.1%). Secondly, reducing the power consumption of compute nodes also implies the power reduction in the cooling system, since less heat is generated by the heat effect. Thus, when we put our sight on the whole system, it can be a great contribution for those systems that cost more than 100 million dollars per year.

Table 6 Enhancement — The Comparison of Hierarchy and others

	Time	Package Power	Memory Power	Total Power	Total Energy
vs. Random	< 0.1%	0.748%	0.341%	0.602%	0.640%
vs. Sequence	< 0.1%	1.555%	0.951%	1.339%	1.346%

6. Conclusions

In this paper, we take process variation and technology problems of hardware into consideration and verified how much impact they have on performance and power. The competition of the top HPC systems becomes more and more vehement, and gigantism of systems will not end, hence, the node level heterogeneity should be aware. Our research focus on the node level performance/power heterogeneity and make advantage of this knowledge to achieve probable power saving in HPC systems.

This paper also introduces a simulator and presents its structure and evaluation on job exclusive mode. In this mode, 1% level power saving was achieved under the FFT2D testing with a maximum usage 67%. More tests will be carried out to evaluate its functions. Moreover, we intend to extend it with job busy mode. Most of the systems that carry out job busy mode have less usage, thus we attempt to make better effects on those systems.

Reference

[1] Andrew J. Younge et al. Providing a Green Framework for Cloud Data Centers. In Handbook on Energy-Aware and Green Computing, page 6, published on December 26, 2011.
[2] Statistics of Performance Development of recorded HPC systems. https://www.top500.org/statistics/perfdevel/, web site on TOP500 Supercomputer Sites.
[3] Auweter A. et al. (2014) A Case Study of Energy Aware Scheduling on SuperMUC. In: Kunkel J.M., Ludwig T., Meuer H.W. (eds) Supercomputing. ISC 2014. Lecture Notes in Computer Science, vol 8488. Springer, Cham.
[4] The 31st TOP500 list to the 47th TOP500 list. Available: https://www.top500.org/lists/top500/
[5] Osman Sarood, Akhil Langer, Abhishek Gupta, Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget, published in SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. (2014)
[6] 宇野 篤也, 末安 史親, 山本 啓二, 肥田 元, 池田 直樹, 辻田 祐一, 消費電力の変動を考慮したジョブスケジューリングの検討, Vol.2017-HPC-161 No.5, 2017/9/19.
[7] Alina Sˆ ırbu, Ozalp Babaoglu, Predicting System-level Power for a Hybrid Supercomputer, published in 2016 International Conference on High Performance Computing & Simulation (2016)
[8] Francesco Fraternali et al. Quantifying the Impact of Variability on the Energy Efficiency for a Next-generation Ultra-green Supercomputer, published in 2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED).
[9] Rong Ge et al. PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 21, NO. 5, pages 658 ~ 671, MAY 2010
[10] Intel official site. Available: https://newsroom.intel.com/press-kits/from-sand-to-silicon-the-making-of-a-chip/
[11] Lide Zhang et al. Process Variation Characterization of Chip-Level Multiprocessors DAC'09, July 26-31, 2009, San Francisco, California, USA
[12] M. Wirnshofer, Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits, Springer Series in Advanced Microelectronics 41, DOI 10.1007/978-94-007-6196-4_2, © Springer Science+Business Media Dordrecht 2013
[13] Enrico Calore et al. Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications, Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe, 2017
[14] Bilge Acun, Phil Miller, Laxmikant V. Kale, Variation Among Processors Under Turbo Boost in HPC Systems, published in ICS '16 Proceedings of the 2016 International Conference on Supercomputing, Article No. 6
[15] Matthew Travers, CPU Power Consumption Experiments and Results Analysis of Intel i7-4820K, Technical Report Series NCL-EEE-MICRO-TR-2015-197.
[16] High Performance Conjugate Gradients (HPCG) official site. Available: http://www.hpcg-benchmark.org/index.html
[17] McCalpin, John D., 1995: "Memory Bandwidth and Machine Balance in Current High Performance Computers", IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December 1995.
[18] T. Hoefler and S. Gottlieb: Parallel Zero-Copy Algorithms for Fast Fourier Transform and Conjugate Gradient using MPI Datatypes. In Recent Advances in the Message Passing Interface (EuroMPI'10), pages 132-141, Springer, ISSN: 0302-9743, ISBN: 078-3-642-15645-8, Sep. 2010

Acknowledgments This work was supported by "Advanced Computational Scientific Program" of Research Institute for Information Technology, Kyushu University. The computation was mainly carried out using the computer facilities at Research Institute for Information Technology, Kyushu University.