Saving 200 kW and $200 K/year by Power-aware Job/Machine Scheduling

Junichi Hikita Akio Hirano Hiroshi Nakashima
Kyoto University
ACCMS North Bldg., Yoshida Hommachi, Kyoto
606-8501, JAPAN
{hikita, hirano, h.nakashima}@media.kyoto-u.ac.jp

Abstract

This paper reports our 3.75-year empirical study on power-aware operations of Kyoto University's supercomputer system. The supercomputer system of 10 TFlops had required about 540 kW on average in its first fiscal year 2004. After that and one-year try-and-error of power efficient operation, we implemented a simple but effective scheduler of jobs and machine powering to improve the per-load power efficiency by up to 39 % and to save 200 kW and $200,000 electric charge in the fiscal year 2006. The power-aware scheduler tries to minimize the number of active nodes in the system of eleven nodes keeping sufficient computational power for given loads. Thus the power-aware scheduler has not degraded, but has significantly improved, the service quality in terms of the average job-waiting time.

1 Introduction

The power consumption issue is now crucial for computer science/engineering and especially for high-performance computing due to a large amount of power requirement to exert high computational performance. As the power consumption of a high-performance system grows larger, the installation and operation of the system become significantly harder due to the increase of the space, the costs of the electricity and cooling facilities, and the charges for electric power.

To reduce the power consumption of a system, we have a large variety of approaches. First, the system itself can be power efficient as BlueGene/L[1] and commodity-base low-power clusters[8, 11]. Second, its compiler and/or run-time system can be smart enough to control DVFS mechanism for the power reduction in a relatively computation-unintensive execution phases[3, 4, 5, 9]. Finally, even if the system and its fundamental software components are not power-aware, we can cut the power of hardware components when the system load is not so high as to require the operation of these components.

For a supercomputer or data center, the first approach is not always feasible because the mission of the center may not be fit to the existing low-power systems. The second one is attractive and easily usable especially when it is taken with automatic DVFS-control software/firmware which CPU venders provide. However, most of more sophisticated power-aware compilers and run-time systems are still at research-level and thus hardly incorporated in the software stack for production run. Moreover, there are still many systems of old microprocessor chips without DVFS which most of power-aware techniques rely on.

In fact, the supercomputer system in Kyoto University, which is the target of the issue discussed in this paper, consists of SPARC 64V processors which are conventional high-end and thus hot ones without DVFS mechanisms. Thus we have to take the third approach, presumably together with most of supercomputer/data centers. Good news for us is that the load of a supercomputer system in a university, at least ours, fluctuates widely depending on seasonal events. For example, the system tends to be heavily loaded in the last few months of an academic year because many students work hard for their degrees, while the load in the first month is usually light. Therefore, if the system has computational power large enough for *high-seasons*, we may save significant electric power in *low-seasons* by powering down some hardware components.

Following the observation above, we designed a simple power-aware job/machine scheduler as discussed in Section 2, and incorporated the scheduling mechanism into our system at the beginning of its third year. Although the scheduler has unique features for good service quality and efficient power reduction, we emphasize neither its novelty nor sophistication. Rather our claim and the primary contribution of this paper are in the result discussed in Section 3. Since our system had operated for two years without the scheduler and have done for one year and nine months with it, we can clearly and quantitatively show its effectiveness

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

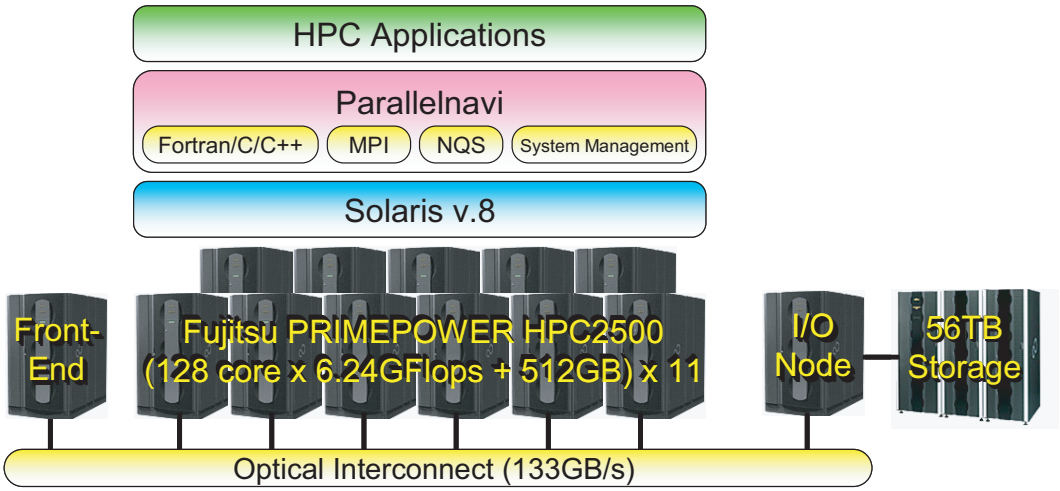


Figure 1. Supercomputer System in Kyoto University

and service quality preservability which should encourage many university supercomputer centers. We also started from this basic achievement toward further improvement of the power efficiency as discussed in Section 4 together with the conclusion of this paper.

2 Power-Aware Scheduling

2.1 Target System

Figure 1 shows the supercomputer system in Kyoto University to which we applied the power-aware scheduling. The major component of the system is the platoon of eleven large scale SMP nodes, Fujitsu PRIMEPOWER HPC 2500, each of which has 128 SPARC 64V single-core processors of 6.23 GFlops and 512 GB shared memory. Thus the peak performance of a node is 798GFlops and that of the platoon is 8.78TFlops by which the system was ranked at 24th in the TOP500 list of June 2004. The nodes, except for one *master* node, can be powered up and down by commands from the master node. The other components are the front-end node of the same architecture for interactive computation and the storage system of 56 TByte controlled by a node also having the same architecture. These components are connected by a high-speed optical network of 133GB/s bandwidth.

The maximum rating of the power consumption of the system is about 600 kW whose major part 385 kW is consumed by the eleven SMP nodes. The effective power consumption is of course less than the maxmimum rating and its measurement is about 450 kW when each node executes Linpack consuming 27 kW and thus 300 kW is consumed by the platoon. The bad news for us is that idling a node saves the power not so much because its processor of 65 W TDP[2] is expected to consume about 50 W and thus the consumption by 128 processors are estimated at 6.4 kW or 24 % of the node's total. Worse, the operating system seems to make idle processors neither halted nor slept but keeps awake with some useless execution. Therefore, in fact, the idle power measurement of a node is not less than 90 % of the peak consumption.

The system is cooled by twelve air conditioners of about 55 kW cooling performance for each. Their coefficient-of-performance (COP) numbers, however, vary in the range from 1.34 to 2.96 depending on the ages of the equipments. Therefore, significant large power of about 300 kW is required to cool the whole system in full operetion resulting in about 750 kW peak power consumption of the total system including the air conditioners. Each of eight conditioners out of twelve has a set of nodes to cool, which may be powered down, as shown in Figure 2, while remaining four are responsible for the cooling of always-powered equipments including the master node #1 which is not shown in the figure. A conditioner may be powered down when all the node in its responsible set are powered down. For example, the CRAC-A may be powered down when the nodes #2 and #3 are done as well, while CRAC-C have to be powered unless all the nodes from #4 to #7 are powered down.

As for the software stack, its bottom is the Solaris v.8 operating system on top of which Fujitsu's all-in-one software package named Parallelnavi resides for automatic parallelized or OpenMP shared-memory programming, MPI programming, NQS-based batch job scheduling and system management. There are two types of job queues, type-s mainly for single-process shared-memory parallel programs and type-d for MPI programs whose processes may have multiple threads. Thus a job in type-s should be assigned on a node while a job in type-d may be split and distributed

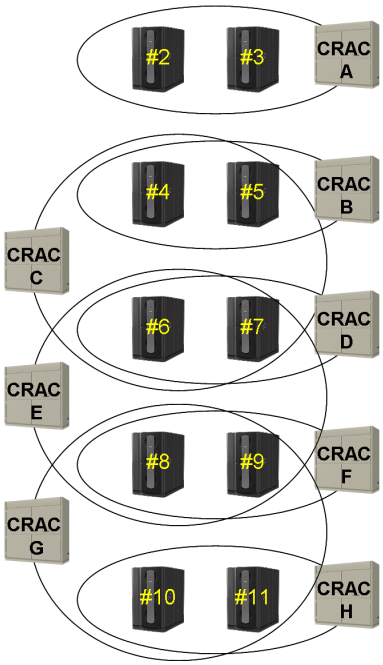


Figure 2. Nodes and Air Conditioners

onto multiple nodes.

The Parallelnavi's system management functions, which run on the master node, include powering up/down of each node, setting resource allocation policy for jobs, and monitoring the status of nodes and job queues. These functions are the fundamental means for our power-aware scheduling.

2.2 Resource Allocation to Jobs

When a job is ready to run with a sufficient number of available resources which the job requires, the job scheduler has to pick resources for the job from those available. Since we have eleven nodes of 128 CPU cores for each, there may be multiple nodes available for a type-s job or multiple node sets available for a type-d job.

If we were unaware of power consumption, a near-optimal strategy could be to assign the type-s job to the node which has the maximum number of available CPU cores. As for the type-d job, its node set could be constructed by picking nodes in descending order of CPU core availability from the top. Since this *distribution* strategy tends to work minimizing the size of resource fragments in the system, we could achieve a small job-waiting time if jobs or their processes consist of a reasonable number of threads, e.g., up to 16. However, the strategy minimizes the chance to emerge idle nodes which may be powered down. For example, it will allocate all eleven nodes to eleven 8-core jobs which could run on a node in whole, and thus will lose the chance of powering down ten nodes.

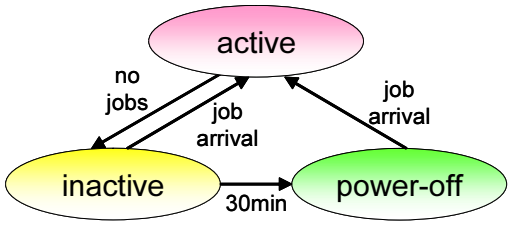


Figure 3. Node State Transition

Thus we chose a strategy at the opposite end of the allocation policy spectrum to *pack* jobs as densely as possible. That is, a type-s job is assigned to the node which has the minimum number of available CPU cores among the nodes able to accommodate the job. The node set for a type-d job is also constructed by minimum-first rule. This type-d job scheduling increases the possibility of inter-node MPI communication, but the bandwidth of the optical interconnection is large enough to cover this additional communications.

The obvious effect of the *packing* strategy is to make idle nodes as many as possible. Thus when the load of the system is so light as to make some nodes idle, we can power down them to save electric energy and cost. The other value-added effect is a high capability of resource allocation to large scale type-s jobs which require many cores up to 128. This feature may compensate the resource fragmentation problem to keep the service quality in terms of the job-waiting time.

2.3 Machine Scheduling and Powering

Figure 3 shows the state transition of a node controlled by our simple machine scheduler. A node stays in one of the following three states; *active* with some jobs running on the node; *inactive* without any jobs but powered up; and *power-off* possibly together with one or more air conditioners responsible for the cooling of the node.

The machine scheduler runs on the master node and continuously monitors other nodes and job queues. When a node finishes its last job and becomes idle, the machine scheduler turns its state to inactive by inhibiting new job assignments to the node. Then the node stays in the inactive state for 30 minutes, or returns to the active state if a new job arrives and cannot run immediately on the active nodes.

After the 30-minute stay in the inactive state without any job assignments, the node is powered down, and the air conditioner responsible for the node is done as well if the node is the last powered one in its responsible set. The 30-minute *buffer* is for keeping the system from too sensitive to short range load fluctuation. Then the node is powered up and moves to the active state when a new job arrives requir-

Table 1. Annual Statistics of Load, Node Powering, Power Consumption, Power Efficiency and Job-Waiting Time

	2004	2005	2006	2007
System Load	1.00	0.80	1.19	1.14
Node Powering Ratio [%]	69.7	56.0	73.5	65.7
Power Consumption [kW]	544	413	500	484
Power Efficiency	1.00	1.20	1.39	1.29
Job-Waiting Time [min:sec]	39:21	6:27	20:58	25:53

ing the node. Powering up a node takes about 45 minutes[1], which is the other reason of the necessity of the 30-minute buffer. If the air conditioner responsible for the node has been powered down, it is also powered up.

If two or more nodes have been powered down and the job arrival requires a part of them, the nodes to be powered up are chosen according to the cooling efficiency. First the machine scheduler tries to find a powered-off node in the node set of an air conditioner which has already been powered up to keep the number of active conditioners. Then if powering up a conditioner is necessary, the node is chosen to minimize the number of conditioners to be powered up. The final tie-break is done by choosing the conditioner with the highest COP rate among those powered-off and a node in its responsible set is powered up. For example, if one of the nodes from #2 to #4 shown in Figure 2 is required to be powered up, the node #4 is chosen because CRAC-B and C for it has already been on. On the other hand, if the nodes from #5 to #7 are also in the set of powering candidates and thus CRAC-A, B and C are off, the node #2 or #3 is chosen because they requires one conditioner CRAC-A only. Finally, if the candidates are the nodes from #2 to #5, the node is chosen from $\{\#2, \#3\}$ or $\{\#4, \#5\}$ depending on the COP superiority of the CRAC-A and B.

3 Empirical Studies

This section shows various statistics obtained from our operation of the system in three years and nine months from April 2004 to December 2007. At first we show the system load fluctuation in Section 3.1. Then the statistics of our node powering opration is shown in Section 3.2, and power consumption and related numbers are presented in Section 3.3. The last two issues are on the possible problems caused by our power-aware scheduling, that is, the possible degradation of the service quality in terms of job-waiting time in Section 3.4, and the possible increase of node failures in Section 3.5.

[1] Almost all time is consumed by self diagnosis procedures which our maintenance contract requires to perform each time of powering up.

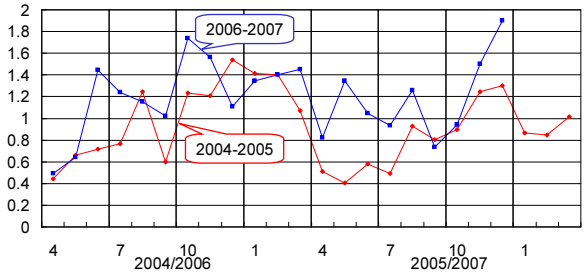


Figure 4. Normalized Monthly System Load

3.1 Load Fluctuation

Figure 4 shows the monthly average load of our system. The load is measured by CPU core usage and normalized by the annual average of the Japanese fiscal (and academic) year 2004. In the following discussion, we refer to a fiscal year from April to March simply as a year.

As the figure clearly shows, the load significantly fluctuates month by month. For example, in the year 2006, the load in October is 3.5 times as heavy as that in April. More complicatedly, the month of heaviest load varies year by year while April is commonly the almost lightest one. That is, the peak months in 2004, 2005 and 2007 (so far) were December which is at the last-one-mile point to master and doctoral dissertations, but 2006 had its highest peak at October and a valley at December. The year 2007 also has incoherence with October, which was busiest in the last year but has almost lightest load in the year, and significantly busy May which was the lightest in 2005 and the third and second lightest in 2004 and 2006 respectively.

The annual average load also fluctuates as shown in the second row of Table 1. The load in 2005 is about 80 % of 2004's while in 2006 we had 1.2 times heavier load. As for 2007, so far the load is a little bit lighter than 2006 but 14 % heavier than the first year.

From these observations, we have to conclude that the system load cannot be anticipated even for monthly level. Therefore if we made a monthly plan of node powering based on the statistics of past years, it should fail, except for April, wasting electric power or degrading service quality.

3.2 Node Powering

Even before we started the operation with the power-aware scheduler described in Section 2 at the beginning of 2006, we had not neglected the power consumption problem. Rather our system administrators had tried to power down manually when the monitoring tool reported the system load was light. Thus as shown in Figure 5, the nodes

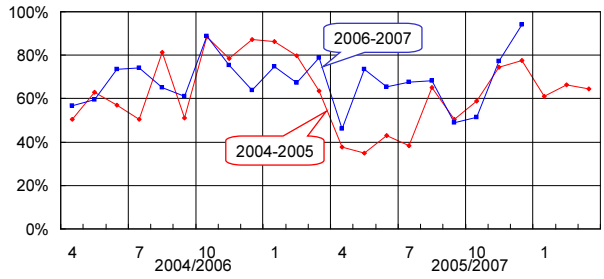


Figure 5. Monthly Node Powering Ratio

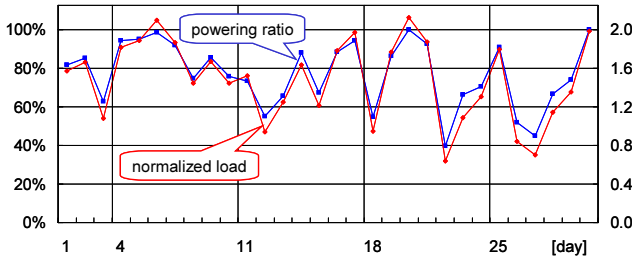
had not been fully powered up even in 2004 and 2005 whose annual average node powering ratios[2] were 70 % and 56 % respectively as shown in the third row of Table 1.

In the first six months of 2004, the chance of powering down was relatively hard to find because the resource was allocated by the *distribution* strategy. Therefore, we powered about 80 % nodes in August 2004, for example, to process the load at the same level in August 2007 in which the powering ratio was less than 70 %. Then we switched the strategy to the *packing* one to have more opportunities to power down nodes. However, the result was not very remarkable because the powering ratio reduction in 2005 almost matched to the load reduction.

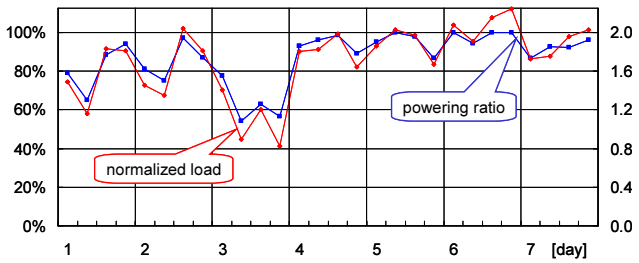
Then we started the power-aware scheduling from 2006 resulting in 74 % powering ratio in the year and 66 % in the first nine months of 2007. Since the load in 2006 and 2007 was 19 % and 14 % heavier than in 2004 respectively, the results prove the effectiveness of the power-aware scheduler.

A closer look of the load and powering statistics shows how our machine scheduler traces the load fluctuation quickly and unfailingly. Figure 6 shows the daily averages of the load and powering in November 2007, their quarter-daily averages of its first week, and the exact percentage of powered nodes together with the hourly load average of the first day. The figure also reveals the difficulty of the load anticipation and manual power control. One might expect that the system may rest in weekends as we do but should be disappointed by unexpectedly heavily loaded Sundays 4th and 25th, and Saturday 17th. It is also tough for system administrators that the load level tends to raise or fall so steeply around midnight in the first week that they could not be asleep if they manually powered nodes up and down tracing the load fluctuation. Thus an automated control is essential not only for the efficient power management but also for the welfare of our administrators.

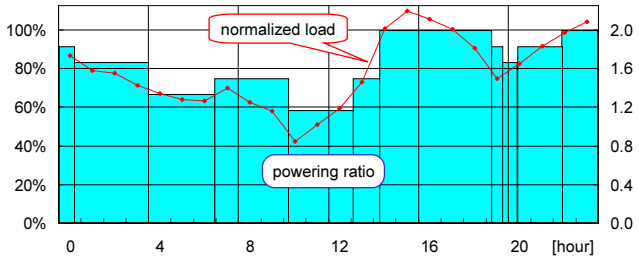
Glancing at the graphs in Figure 6, one might apprehend that nodes should be powered up and down too frequently, for example, several times in a day. However, in November 1, for example, only one node is powered up, down then up, and the other nodes are powered up and down once at most. This relatively infrequent powering is evidenced by the long term statistics shown in Figure 7. In the figure, the average off-time (blue or dark) and on-time (pink or light) of each power-manageable node are illustrated by four bars for the years 2005, 2006 and 2007 and the three-year average from left to right[3]. The figure also shows the times averaged in ten nodes by the rightmost four-bar group which indicates that the average powering cycle of a node is about two days, one day off and one day on approximately. Even for the most frequently powered node #3, the cycle is longer than one day. Although these observations does not assure us that the cycle is sufficiently long to keep



(a) monthly statistics



(b) weekly statistics of Nov. 1–7



(c) daily statistics of Nov. 1

Figure 6. Monthly, Weekly and Daily Statistics of Normarized Load and Powering Ratio in Nov. 2007

[2]The node powering ratio is defined as the percentage of the uptime averaged in all the nodes and given period.

[3]Although we have the complete log for the number of powered nodes, in 2004 we did not keep the log to know the time and node ID of each powering event.

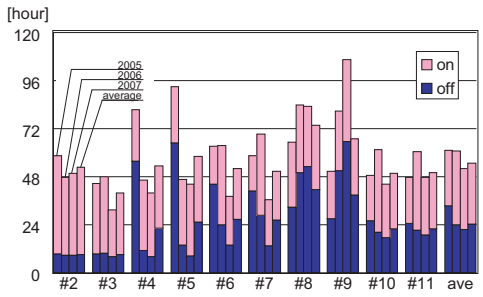


Figure 7. Average Power-On/Off Time of Nodes

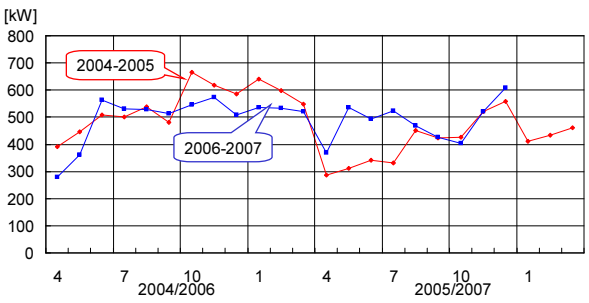


Figure 8. Monthly Average Power Consumption

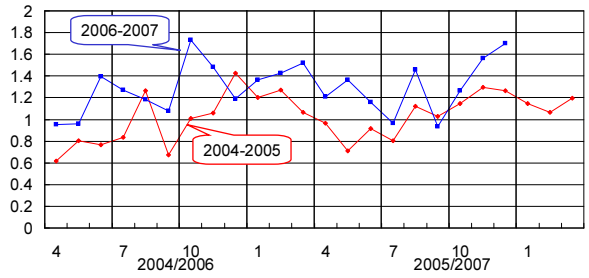


Figure 9. Normalized Monthly Power Efficiency

nodes from failures caused by powering, they at least clarify we have not performed powering operations harder than an earth-friendly one for office PCs powered up each morning and down each evening. The analysis of the relationship between powering frequency and failures is given in Section 3.5.

Another interesting fact observed from the figure is the shift of *negative preference* of powering from the nodes #4 and #5 in 2005 to #8 and #9 in 2006 and 2007. That is, our administrators seemed to hesitate powering #4 and #5 up in 2005 following their own preference. In the following years, on the other hand, our power-aware scheduler has had the tendency to keep #8 and #9 powered-off because they are cooled by three air conditioners which are at the bottom of the COP ranking.

3.3 Power Consumption and Efficiency

The monthly average of power consumption shown in Figure 8 more clearly evidences the effectiveness of the power-aware scheduling. In almost all months of 2006, the power consumption was less than that in 2004 and was in reverse of the load statistics. The annual average of 2006 is 500 kW and of course less than 544 kW of 2004 as shown in the fourth row of Table 1, but this is surprising because the node powering ratio in 2006 is higher than in 2004[4]. This power reduction results from the node powering priority taking the cooling power and efficiency into acount. In fact, the per-node power consumption in 2006 is 87 % of that in 2004.

As a whole, the power efficiency was significantly improved by the power-aware scheduling as shown in Figure 9 in which normalized ratios of load to power consumption are plotted using the annual average of 2004 as the baseline. That is, the power efficiency E of a given period is defined by

$$E = \frac{L}{L_{2004}} \cdot \frac{P_{2004}}{P}$$

where L and L_{2004} are the average load factors in the period and 2004 respectively, and P and P_{2004} are the average power consumption in the period and 2004 respectively. As for the annual average, 2006's power efficiency was improved by 39 % from 2004 as shown in the fifth row of Table 1. This means the average power consumption in 2006 could be 1.39 times as large as its actual 500 kW resulting in about 700 kW if we had operated the system as in 2004. Since the total energy consumption in 2006 was 4.3 TWh, it could be 6.0 TWh if we had not applied the power-aware scheduler to our system. Thus we can conclude that we saved about 200 kW power consumption and 1.7 TWh energy in 2006 or about $200,000 cost charged by $0.114/kWh rate[5].

[4]The power consumption in October 2006 is also surprising even for the authors because of the following. It was the second busiest month with the second highest powering ratio in our operation history but its average power consumption is less than, for example, November in the year whose powering ratio is 10 % less than October. Although the reason could be found in environmental factors affecting cooling efficiency, we could not find any data to explain this anomaly.

[5]Calculated by dividing our actual 12 JPY/kWh rate by 105 JPY/USD exchange rate.

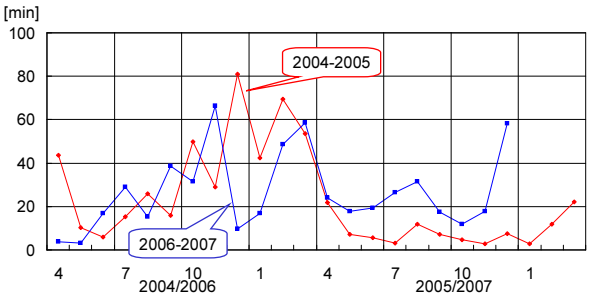


Figure 10. Monthly Average Job-Waiting Time

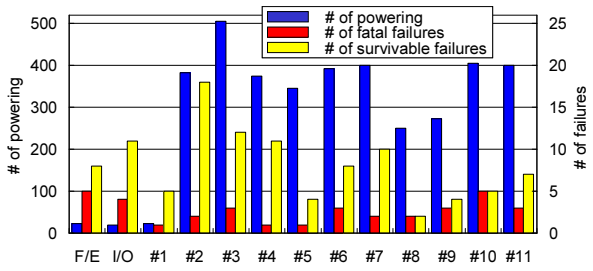


Figure 11. Number of Powering and Failures of Nodes

3.4 Service Quality

As discussed in Section 2, the packing strategy might produce resource fragments in nodes to degrade the capability of resource allocation to arriving jobs. The relatively long time for node powering up is also a potential cause to lengthen job-waiting time. Thus our concern was how the power-aware scheduling had affected to the service quality in terms of job-waiting time.

As shown in Figure 10, monthly averages of job-waiting time in 2006 and 2007 did not worsen from 2004 values. Rather, annual averages of these years shown in the sixth row of Table 1 are about half and two third of 2004's average respectively. Since the quick service in 2005 is considered not for the difference in the scheduling but for the light load in the year, we can conclude that the power-aware scheduling is not harmful to the service quality.

3.5 Powering and Node Failure

Another concern about our power-aware scheduling is that bi-daily powering of nodes could significantly increase the failure rate of them. To examine whether the suspicion of power-aware but machine-breaking scheduling is based on any evidence, we enumerated the numbers of powering and hardware failures of each node from 2005 to 2007[6]. The result is shown in Figure 11 which has three bars for each node not only power-manageable but also (almost) always-powered, i.e., the front-end, I/O and master (#1) nodes. The leftmost bar for a node is the number of powering up in three years (left axis), while other two are for node failures (right axis) categorized by their fatality. That is, the second bar represents the number of failures causing machine down, while the rightmost is for those survivable such as the correctable memory errors exceeding the predefined ignorable count.

[6]As noted in Section 3.2, per-node powering log in 2004 is not preserved unfortunately. The failure report in the year is retained of course but its significantly large fraction is for early failures.

It is hard to find any clear relationships between the numbers of powering and failures. The node with the largest number of fatal failures, five, is not only in power-manageable ones (#10) but the front-end node has also suffered with the same number of fatal failures. The largest number of survivable failures has occured in the node #2 but its powering count is on the same level of other nodes and is clearly less than that of #3 whose survivable failure count is similar to those of the nodes #4 and #7 and the always-powered I/O node.

Therefore, it should be allowed to conclude that our power-aware scheduling does not directly and immediately affect the hardware reliability so far. Although it is unknown whether the scheduling shorten the lifetime of nodes, we are all right if they will be healthy until the end of our four-year contract.

4 Concluding Discussion

This paper described our power-aware job/machine scheduling applied to Kyoto University's supercomputer system. The scheduling mechanism is so simple that any supercomputer systems may be equipped with it without any special low-power technologies. The importance of our work is in our empirical studies by which we showed the effectiveness of the power-aware scheduling with various statistical data obtained from our operations in three years and nine months without and with the scheduling. The most impressive result is the 39 % power efficiency improvement achieved in 2006, which saved about 200 kW power consumption and $200,000 cost for annual electric power.

Our success with the simple power-aware scheduling, however, does not lead us to the simple-is-best conclusion. Rather it gives us an expectation that more sophisticated mechanisms will achieve much more power reduction and/or much less negative effects to the service quality. For example, exploiting socket/core-level DVFS and/or sleeping mechanisms, we will be able to have a fine-grain and quick-

response power-aware machine scheduler. In fact, we will replace the fat-node system with a thin-node based one with 416 quad-socket nodes of quad-core processors with core-level power-aware mechanisms in June 2008[10] and will try more sophisticated power-aware operation with it.

The other approach of sophistication should be found in the system load prediction to power up/down nodes more appropriately and adaptively. As discussed in Section 3.1, the load amount of our system is hardly predictable at least from a macroscopic view, unlike the case of data centers[7]. However, it does not deny the possibility of microscopic prediction from various viewpoints. For example, a short range prediction of daily or weekly fluctuation may be possible if we precisely analyze running and arriving jobs with respect to, for example, their requiring resources, programs and input data, owner users and so on. Thus at first we will investigate the predictability using detailed logs obtained from our four year operations.

The logs should be also useful to evaluate a variation of power-aware techniques and their parameters without applying them to our system. The effectiveness of our power-aware scheduling is confirmed by an unintentionally established test-bed, our two year operation without it and the following one year and nine months with it. This means that we had paid a high tuition for two years to learn its effectiveness. Thus we need to do a what-if type simulation[6] using the logs next time to learn how we have to do and how large reduction we will achieve.

Acnowledgements We would like express our appreciation to the technical staff in Fujitsu Ltd. for their support to our operation and cooperation by which we obtained invaluable statistics. This research is partly supported by Grant-in-Aid of Scientific Research #19024041 of MEXT Japan, and the Global COE Program of Kyoto University entitled "Informatics Education and Research for Knowledge-Circulating Society" sponsored by MEXT Japan.

References

[1] N. R. Adiga et al. An overview of the BlueGene/L supercomputer. In *Proc. Supercomputing 2002*, Nov. 2002.

[2] H. Ando et al. A 1.3-ghz fifth-generation sparc64 microprocessor. *IEEE J. Solid-State Circuits*, 38(11), Nov. 2003.

[3] H. Dietz and W. Dieter. Compiler and runtime support for predictive control of power and cooling. In *Proc. 2nd WS. High-Performance, Power-Aware Computing (in Proc. IPDPS 2006)*, Apr. 2006.

[4] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based optimization of power-performance by using dynamic voltage scaling on a PC cluster. In *Proc. 2nd WS. High-Performance, Power-Aware Computing (in Proc. IPDPS 2006)*, Apr. 2006.

[5] C. Hsu and W. Feng. A power-aware run-time system for high-performance computing. In *Proc. ACM/IEEE SC|05 Conf.*, Nov. 2005.

[6] R. Ito. An experimental study of NQS parameter tuning method for improvement of the system usage efficiency. In *Proc. Intl. WS. Automatic Performance Tuning*, Sept. 2007.

[7] B. Khargharia et al. Autonomic power and performance management for large scale data centers. In *Proc. Next Generation SoftwareWS. (in Proc. IPDPS 2007)*, Mar. 2007.

[8] H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka, D. Takahashi, and Y. Hotta. MegaProto: 1 TFlops/ 10 kW rack is feasible even with only commodity technology. In *Proc. ACM/IEEE SC|05 Conf.*, Nov. 2005.

[9] K. C. R. Ge, X. Feng. Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In *Proc. ACM/IEEE SC|05 Conf.*, Nov. 2005.

[10] T2K open supercomputer alliance. http://www.open-supercomputer.org/.

[11] M. Warren, E. Weigle, and W. Feng. High-density computing: A 240-node Beowulf in one cubic meter. In *Proc. Supercomputing 2002*, Nov. 2002.