BSLD Threshold Driven Power Management Policy for HPC Centers

Maja Etinski[†]
maja.etinski@bsc.es

Julita Corbalan[†,‡]
julita.corbalan@bsc.es

Jesus Labarta[†,‡]
jesus.labarta@bsc.es

Mateo Valero[†,‡]
mateo.valero@bsc.es

[†]Barcelona Supercomputing Center
Jordi Girona 31, 08034 Barcelona, Spain

[‡]Department of Computer Architecture
Technical University of Catalonia

Abstract

In this paper, we propose a power-aware parallel job scheduler assuming DVFS enabled clusters. A CPU frequency assignment algorithm is integrated into the well established EASY backfilling job scheduling policy. Running a job at lower frequency results in a reduction in power dissipation and accordingly in energy consumption. However, lower frequencies introduce a penalty in performance. Our frequency assignment algorithm has two adjustable parameters in order to enable fine grain energy-performance trade-off control. Furthermore, we have done an analysis of HPC system dimension. This paper investigates whether having more DVFS enabled processors for same load can lead to better energy efficiency and performance. Five workload traces from systems in production use with up to 9 216 processors are simulated to evaluate the proposed algorithm and the dimensioning problem. Our approach decreases CPU energy by 7%- 18% on average depending on allowed job performance penalty. Using the power-aware job scheduling for 20% larger system, CPU energy needed to execute same load can be decreased by almost 30% while having same or better job performance.

1. Introduction

Processor power consumption presents one of the major components of the total system power consumption. DVFS (Dynamic Voltage Frequency Scaling) technique is commonly used to manage CPU power. It supports a certain set of frequency-voltage pairs called gears. Running a processor at lower frequency/voltage results in lower power/energy consumption. In HPC (High Performance Computing) systems, single application power control has been explored applying DVFS to parallel scientific applications ([12, 18, 15]). Its application to large scale parallel systems at the system level has been left unexplored. To the best of our knowledge there is only one attempt to integrate frequency scaling into parallel job scheduling and it is only for scheduling of bag-of-tasks applications ([16]) although power reduction is seen as one of the new challenges of parallel job scheduling ([5]).

In this paper, the well established EASY backfilling policy ([21]) is used as a base job scheduling policy. Parallel job scheduling policy determines when a job submitted to the HPC center will be run. We propose a power-aware scheduler that should additionally determine which frequency the job will be run at. Running certain jobs at lower frequency saves energy. Unfortunately, it decreases overall performance. Besides the penalty that affects the job which has been executed at lower frequency, other jobs can experience longer wait time. Two adjustable parameters, $BSLDthreshold$ and $WQthreshold$, are introduced to control the both sources of performance decrease. Altough our work extends the EASY backfilling policy, the frequency scaling algorithm can be applied with any parallel job scheduling policy.

As it has been already remarked ([6, 2]), running an application at lower frequency on more processors can be more energy efficient than running it at the highest CPU frequency on less processors. Furthermore, in that way better

978-1-4244-6534-7/10/$26.00 ©2010 IEEE

energy efficiency can be achieved with no penalty in performance. We have examined energy consumption and job performance of enlarged DVFS systems. Since our jobs are rigid we have used original job sizes.

Main contributions of this paper are:

- We have proposed and evaluated a new CPU frequency assignment algorithm for HPC structures.
- A mechanism that allows performance-energy trade-off control has been designed and analyzed.
- A new simulation infrastructure has been developed making it easy to analyze different system and scheduling policy parameters.
- We have analyzed the impact of system size on energy savings and job performance.

The proposed policy and the system size impact are evaluated based on simulations of five workloads of systems in production use. Our approach decreases CPU energy by 7%- 18% on average depending on allowed job performance penalty. Applying the same frequency scaling algorithm to 20% larger system, CPU energy needed to execute same load can be decreased by almost 30% while keeping the original performance.

The rest of the paper is organized as follows. The next section describes the frequency assignment algorithm. The simulation framework is explained in Section 3. Section 4 describes power and execution time models implemented in the simulator. Results are given in Section 5. An overview of related work is exposed in Section 6. The last section concludes our work.

2. Power-Aware Parallel Job Scheduling

2.1 EASY Backfilling

We have upgraded the EASY backfilling policy ([21]) in a power-aware manner. Backfilling policies demand that users submit an estimation of job run time. With the EASY backfilling jobs are executed in FCFS order except when the first job in the wait queue can not start because there is no enough resources. A job is executed before previously arrived ones if its execution does not delay the first job in the queue. The start time of the first job in the queue is determined based on requested times of already running jobs. A job executed before previously arrived ones is a backfilled job. Backfilled jobs usually request short execution time or a small number of processors. Rescheduling of all queued jobs is done when a job finishes earlier than it has been expected according to its requested time.

2.2 Frequency Assignment

In order to control performance while determining job frequency we regard BSLD (Bounded Slowdown) metric. BSLD is widely used metric of user satisfaction. It gives the ratio between the time spent in the system and the job run time:

$$BSLD = max(\frac{WaitTime + RunTime}{max(Th, RunTime)}, 1) \quad (1)$$

where $WaitTime$ and $RunTime$ are time that the job spent waiting on the execution and the execution time. Th is a threshold used to avoid impact of very short jobs on the average value. In our experiments it is set to 600 seconds as HPC jobs shorter than 10 minutes can be assumed to be very short jobs.

According to our algorithm a job will be run at a lower frequency if its predicted BSLD at the lower frequency is less than previously set $BSLDthreshold$. Predicted BSLD is computed in the following way:

$$PredBSLD = max(\frac{WT + RQ * Coef(f)}{max(Th, RQ)}, 1) \quad (2)$$

where WT is the job wait time according to the current schedule. Requested time RQ presents a run time estimate submitted by the user and $Coef$ is a time penalty function that depends on CPU frequency. How frequency scaling affects execution time is described in Section 4.

With the EASY backfilling a job can be scheduled in two manners. If the job is the head of the wait queue it is allocated with ***MakeJobReservation(J)***. Depending on current resource availability the job will be sent to execution immediately or a reservation will be made for it. The other way to schedule a job is with ***BackfillJob(J)*** function. It is called when there is already a job with a reservation. ***BackfillJob(J)*** tries to find an allocation for the job such that the reservation is not violated. ***MakeJobReservation(J)*** and ***BackfillJob(J)*** algorithms are shown in Figure 1 and Figure 2 respectively.

The scheduler iterates starting from the lowest available CPU frequency trying to schedule a job such that its predicted BSLD is lower than $BSLDthreshold$. If it can not be scheduled at the lowest frequency, the scheduler tries with higher ones. The job will be run at reduced frequency only if there are no more than $WQthreshold$ jobs in the wait queue (jobs waiting on execution). Otherwise it will be run at the highest frequency F_{top}. This parameter controls the impact of frequency scaling on jobs that will be executed.

```
MakeJobReservation(J)
if ($WQsize \leq WQthreshold$) then
    for $f = F_{lowest}$ to $F_{top}$ do
        Alloc = findAllocation(J,f);
        if (satisfiesBSLD(Alloc, J, f)) then
            schedule(J, Alloc);
            break;
        end if
    end for
else
    Alloc = findAllocation(J,$F_{top}$)
    schedule(J, Alloc);
end if
```

Figure 1. Making a job reservation

```
BackfillJob(J)
if ($WQsize \leq WQthreshold$) then
    for $f = F_{lowest}$ to $F_{top}$ do
        Alloc = TryToFindBackfilledAllocation(J,f);
        if (correct(Alloc) and satisfiesBSLD(Alloc, J, f)) then
            schedule(J, Alloc);
            break;
        end if
    end for
else
    Alloc = TryToFindBackfilledAllocation(J,$F_{top}$)
    if (correct(Alloc) and satisfiesBSLD(Alloc, J,$F_{top}$)) then
        schedule(J, Alloc);
    end if
end if
```

Figure 2. Backfilling a job

3. Simulation Framework

3.1 The Simulator

We have upgraded the Alvio simulator ([9]) to support DVFS enabled clusters and the power-aware scheduler. Alvio is an event driven C++ simulator that supports various backfilling policies. A job scheduling policy interacts with a resource selection policy which determines how job processes are mapped to the processors. It governs allocation search in *findAllocation* and *TryToFindBackfilledAllocation* functions. First Fit is used as the resource selection policy in the simulations.

3.2 Workloads

Cleaned traces of five logs from Parallel Workload Archive ([13]) are used in the simulations. A cleaned trace does not contain flurries of activity by individual users which may not be representative of normal usage. Table 1 summarizes workload characteristics. The number in workload name presents the number of processors that the system comprises of. We have simulated 5000 job part of each workload. The parts are selected so that they do not have many jobs removed. Simulated workload parts and the average BSLD values when no DVFS is used are given in the table.

The CTC log contains records for IBM SP2 located at the Cornell Theory Center. The log presents a workload with many large jobs but with relatively low degree of parallelism. SDSC and SDSC-Blue logs are from the San Diego Supercomputing Center. The SDSC workload has less sequential jobs than the CTC workload while run time distribution is very similar. In the SDSC-Blue workload there are no sequential jobs, to each jobs is assigned at least 8 processors. LLNL-Thunder and LLNL-Atlas workloads contain several months worth of accounting records in 2007 from systems installed at Lawrence Livermore National Lab. Thunder was devoted to running large numbers of smaller to medium jobs while Atlas cluster is used for running large parallel jobs. More information about workloads can be found in Parallel Workload Archive ([13]).

Workload - #CPUs	Jobs (K)	Avg BSLD
CTC - 430	20 - 25	4.66
SDSC - 128	40 - 45	24.91
SDSCBlue - 1152	20 - 25	5.15
LLNLThunder - 4008	20 - 25	1
LLNLAtlas - 9216	10 - 15	1.08

Table 1. Workloads

4 Power and Time Models

CPU power consists of dynamic and static power. Dynamic power depends on the CPU switching activity while static power presents various leakage powers of the MOS transistors.

The dynamic component equals to:

$$P_{dynamic} = ACfV^2 \tag{3}$$

where A is the activity factor, C is the total capacity, f is the CPU frequency and V is the supply voltage. In our model we assume that all applications have same average activity factor. Different activity factors are used for idle CPUs and for CPUs running a job. The activity of a running processor is assumed to be 2.5 times higher than the activity of an idle processor. The value is based on measurements from related work ([4, 14]).

According to [1] static power is proportional to the voltage:

$$P_{static} = \alpha V \tag{4}$$

where the parameter α is determined as a function of the static portion in the total CPU power of a processor running at the top frequency. All the parameters are platform dependent and adjustable in configuration files. In our experiments static power makes 25% of the total active CPU power at the highest frequency.

We have used DVFS gear set given in Table 2.

Frequency (GHz)	0.8	1.1	1.4	1.7	2.0	2.3
Voltage (V)	1	1.1	1.2	1.3	1.4	1.5

Table 2. DVFS gear set

Energy modeled in our simulations assumes that idle processor consume power corresponding to the lowest frequency and the idle processor activity factor. In this case according to the model, an idle processor consumes 21% of the power consumed by a processor executing a job at the highest frequency.

Energy consumed by a workload when assumed that idle CPUs dissipate no power is computed as well. We refer to this energy as computational energy. There are processors that dissipate almost no power in low power modes. Furthermore, design of a whole system that consume negligible power while idling has been proposed ([19]). Exploring computational energy gives an insight into full potential of system enlarging and frequency scaling.

How frequency scaling affects computation time depends on the application memory boundedness and its communication intensity. The simulator uses the β execution time model from [12]. It is based on the following equation:

$$T(f)/T(f_{max}) = \beta(f_{max}/f - 1) + 1 \tag{5}$$

where $T(f)$ is a job run time at CPU frequency f, $T(f_{max})$ is the job run time at the top frequency $fmax$. It works for both sequential and parallel applications as there is no difference whether the CPU is waiting for data from the memory system or from an another node. $\beta = 1$ means that halving frequency doubles the execution time whereas $\beta = 0$ means that a change in frequency does not affect the execution time. β is assumed to be 0.5 according to measurements from [8].

5. Results

Energy results reported in this section are normalized with respect to the energy values when no DVFS is used. As a measure of performance, BSLD averaged over all simulated jobs is considered. When frequency scaling is applied to a job its BSLD is computed in the following way:

$$BSLD = max(\frac{WaitTime + PenalizedRunTime}{max(Threshold, RunTime)}, 1) \tag{6}$$

where $PenalizedRunTime$ is the job run time at the reduced frequency.

5.1 Original System Size

First we focus on the original size systems. The frequency assignment algorithm has two parameters $BSLDthreshold$ and $WQthreshold$. We have tested three values for $BSLDthreshold$: 1.5, 2 and 3. Four different values are used for $WQthreshold$. 0 means no DVFS will be applied if there is a job waiting on execution. Then, we have tested two less restrictive thresholds for the wait queue size, 4 and 16 jobs. The last used threshold sets no limit on the wait queue size. It means that CPU frequency is assigned only based on the predicted BSLD.

The reduction in CPU energy consumption achieved by the frequency scaling is presented in Figure 3. The first graph presents reduction in computational CPU energy (idle processors do not dissipate power). The second graph gives results assuming that idle processors consume energy according to the model described in Section 4. As we can remark, there is almost no difference between the two graphs as each presents values normalized to their corresponding original values.

All workloads except SDSC shows an energy decrease of about 10% or more depending on the thresholds used. For the least restrictive combination of parameters ($BSLDthreshold$ = 3 and $WQthreshold$ = NO LIMIT) their savings in computational energy are up to 22%. The SDSC workload has the worst original performance, its average BSLD without frequency scaling is 24.91. Hence the proposed policy with used $BSLDthreshold$ values can not lead to an energy decrease.

For a fixed $BSLDthreshold$ value increasing the wait queue limit gives frequency schedule that results in lower energy. But for a fixed wait queue limit an increase in $BSLDthreshold$ does not necessary result in higher energy savings as one might expect. For instance, the LLNLThunder workload saves 8.95% of computational energy for $BSLDthreshold$ = 1.5 and $WQthreshold$ = 4 while for the same $WQthreshold$ and $BSLDthreshold$ = 2 it saves 3.79% of computational energy. In the first case 1219 jobs are run at lower frequency while that number in the second case is 854. Higher $BSLDthreshold$ can lead to less reduced jobs in total due to an increase in wait time of jobs that arrive after frequency scaling. In Figure 4 we can see the number of jobs run at lower frequency for each workload and parameter combination.

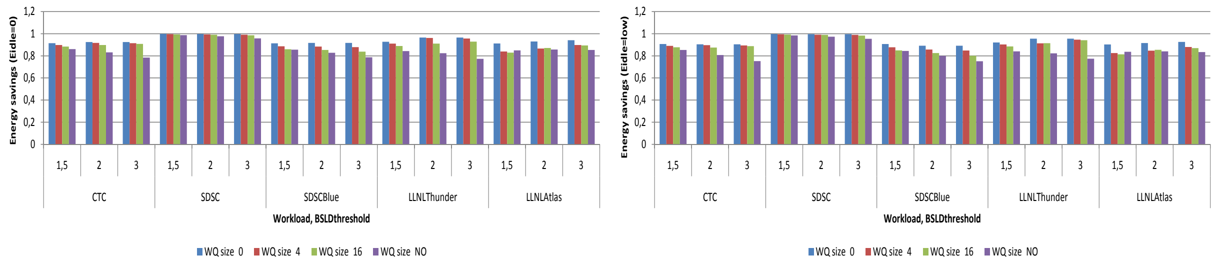


Figure 3. Normalized energy for original system size

Besides the number of jobs run at reduced frequency, their size and frequency used are important as well. For instance, when using the policy for $BSLDthreshold = 2$ and $WQthreshold$ = NO LIMIT for the SDSCBlue workload there are 2778 jobs run at lower frequency. When the parameters are $BSLDThreshold = 3$ and $WQthreshold$ = NO LIMIT there are 2654 jobs executed at lower frequency but in this case greater energy savings are achieved.

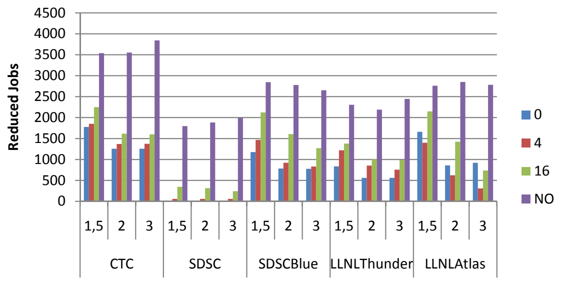


Figure 4. Number of jobs run at reduced frequency

Average BSLD values are given in Figure 5. As shown in Table 1 the average BSLD value varies significantly for different workloads. The SDSC workload has the highest original average BSLD 24.91. The best one is LLNLThunder's average BSLD which is equal to 1. Majority of LLNLThunder jobs are shorter than the threshold from the formula (1), hence their BSLD is 1.

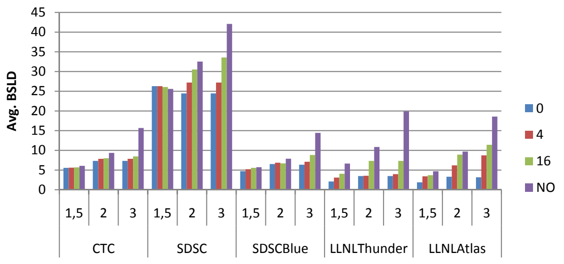


Figure 5. Average BSLD

The most aggressive parameter combination $BSLDthreshold = 3$ and $WQthreshold$ = NO LIMIT penalizes the most the average BSLD but it gives the highest energy savings. However, penalty in performance is not always directly proportional to energy savings. For example, $BSLDthreshold = 1.5$ and $WQthreshold$ $= 0$ threshold combination for LLNLAtlas gives better energy and performance than $BSLDthreshold = 2$ and $WQthreshold = 0$.

The penalty in performance for original system size is significant, especially for more aggressive threshold combinations. Frequency scaling affects performance of a job to which it is applied. It can additionally affect jobs that will be executed after the reduced job as their wait time can be increased.

Job wait time in seconds during a part of the SDSCBlue workload execution is shown in Figure 6. The lower line presents wait time without frequency scaling and the upper one shows wait time for $BSLDthreshold = 2$ and $WQthreshold = 16$ parameter combination. It can be observed that wait time with frequency scaling is much higher than without it.

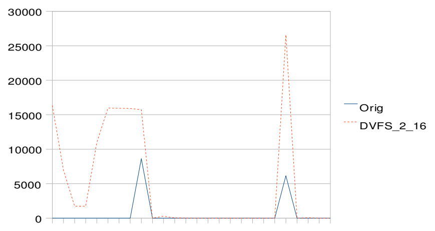


Figure 6. Zoom of SDSCBlue wait time behavior

5.2 Increased System Size

As it has been already mentioned, we have scheduled the same workloads with the power-aware scheduler on enlarged systems. The goal has been to see whether applying

DVFS to enlarged systems can reduce operating costs while having same or better performance. A larger system should amortize the increase in job wait time due to frequency scaling. We have performed a set of experiments with configurations that target the most conservative and the most aggressive scenarios: $WQthreshold$ = 0 and $WQthreshold$ = NO LIMIT. $BSLDthreshold$ has been set to the medium used value 2.

Figure 7 gives energies normalized with respect to the case of original system without frequency scaling for $WQsize$ = 0. $WQsize$ = NO LIMIT energy savings are presented in Figure 8. X-axis shows various system sizes ranging from the original size to 125% increase in system size. Now, two energy scenarios show different behavior. Logically, computational energy decreases with system dimension increase. Larger system gives more opportunity for frequency reduction as BSLD decrease due to shorter wait times. Increasing system size by 20% and using the power-aware scheduling is possible to decrease computational energy more than 25%. In the other energy scenario, due to the increased number of processors, savings are slightly lower and there is a point after which further increase in system size results in higher energy consumption.

Figure 9 shows the impact of the power-aware scheduling on job performance when applied to enlarged systems. While applying the power-aware scheduling an additional increase in system size always gives an improvement in performance although there are more jobs running at lower frequency. LLNLAtlas and LLNLThunder workloads have perfect or almost perfect performance without frequency scaling. Therefore, system enlarging can not improve their original performance. However it can save energy with minimal penalty in performance. CTC, SDSC and SDSCBlue workloads with system enlarging and frequency scaling achieve better performance than the original one. For instance, SDSCBlue with the power-aware scheduler needs only 10% of system size increase to show an improvement in job performance compared to the one without frequency scaling.

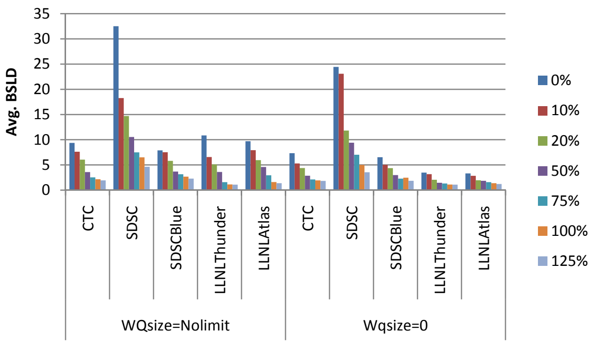


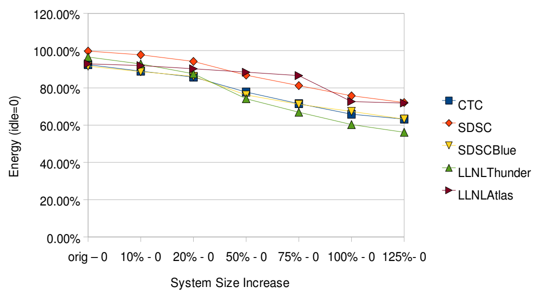
Figure 9. Average BSLD for enlarged systems

Table 3 gives the average wait time in seconds for the following scheduling/system configurations: original system size without frequency scaling, original system size with the power-aware scheduling for $BSLDthreshold$ = 2 and two values of $WQsize$ (0 and no limit), 50% enlarged system with the power-aware scheduling for the same parameters as in the previous case. The wait time values explain BSLD behavior.

6. Related Work

The first group of related works presents works at the application level. Power profiling of parallel applications under different conditions is done in some works ([6, 14]). Although they just report power and execution time on specific platforms for different gears or numbers of nodes, they give a valuable insight in relations between CPU frequency, power and execution time. There are power reduction systems based on previous application profiling ([23, 7, 11]). Several runtime systems that apply DVFS in order to reduce energy consumed per an application are implemented ([12, 18, 15]). These systems are designed to exploit certain application characteristics like load imbalance of MPI applications or communication-intensive intervals. Therefore, they can be applied only to certain jobs.

The second group of works target system level power management of large scale systems. Lawson et al aim to decrease supercomputing center power dissipation powering down some nodes ([17]). The EASY backfilling is used as the job scheduling policy. In this manner BSLD is affected seriously in cases of high load. Two policies are proposed to determine the number of active nodes of the system. The first policy proposed is two level policy that fluctuates the number of active processors between the maximum number of processors and a system-specific minimum number of processors. In the presence of fluctuating workload conditions, the two level policy does not behave well. Online simulation policy executes multiple online simulations assuming different numbers of active processors. The system chooses the lowest number of active processors whose computed average slowdown satisfies the predefined service level agreement (SLA). An empirical study on powering down some of system nodes is done ([10]). A resource selection policy used to assign processors to a job is designed in order to pack jobs as densely as possible and accordingly to allow powering down unused nodes. Kim et al propose a power aware scheduling algorithm for bag-of-tasks applications with deadline constraints on DVFS enabled clusters ([16]). It gives a frequency scaling algorithm for a specific type of job scheduling with deadline constraints that is not common in HPC centers. There are works on en-



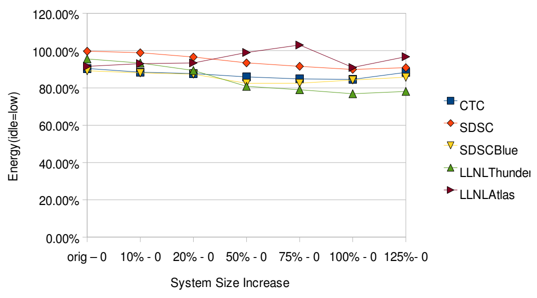
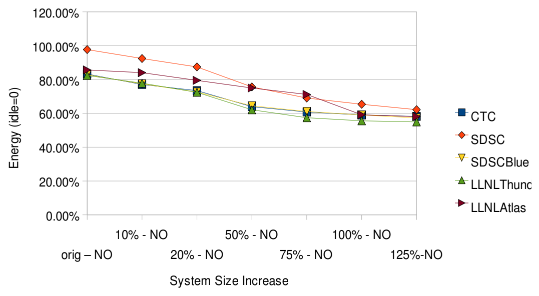


Figure 7. Normalized energies of enlarged systems - WQ size = 0



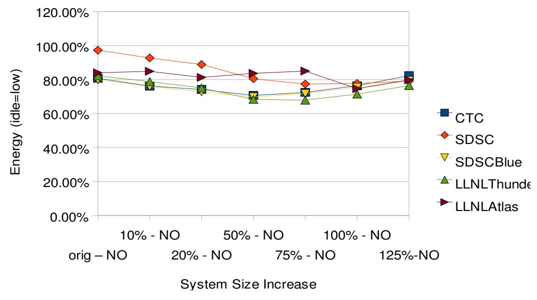


Figure 8. Normalized energies of enlarged systems - NO WQ limit

ergy efficiency of server clusters. Fan et al explore the aggregate power usage characteristics of large collection of servers ([3]). The authors also investigate possibility of energy saving using DVFS that is triggered based on CPU utilization. Elnozahy et al. propose policies for server clusters that adjust the number of nodes online as well as their operating frequencies according to the load intensity ([20]). Pinheiro et al also decrease power consumption by turning down cluster nodes under low load ([22]). Since shutting a node of their system takes approximately 45 seconds and bringing it back up takes approximately 100 seconds, it is not recommended to simply shut down all unused nodes.

7. Conclusions and Future Work

In this paper, we have proposed power-aware parallel job scheduling based on DVFS technique. It assigns CPU frequency to a job regarding its performance and the number of jobs waiting on execution. The policy has been evaluated for five workload logs from large systems in production used. Our approach decreases CPU energy by 7%- 18% on average depending on allowed job performance penalty. As frequency scaling decreases job performance because of an increase in job wait time, we have examined potentials of system enlarging. It has been concluded that it is possible to reduce energy and improve job performance increasing system size and applying the power-aware scheduling. For instance, an increase of 50% in systems size can give much better job performance and up to 35% reduction in computational energy.

At current stage, frequencies are assigned at the scheduling time, one for whole execution. We will add a possibility to dynamically increase frequencies of jobs running at lower frequencies when there are too many jobs waiting on execution.

The β parameter from the power model depends on job CPU boundedness and communication intensity. In this work we have assumed an average value to be same for all jobs. In future we plan to perform an analysis of the β parameter that would allow modeling of different job potentials to exploit DVFS.

8. Acknowledgments

This work has been supported by the Spanish Ministry of Science and Education under contract TIN200760625C0201 and by the IBM/BSC MareIncognito project under the grant BES-2005-7919, and by the Generalitat de Catalunya (2009-SGR-980)

References

[1] J. Butts and G. Sohi. A static power model for architects. *Microarchitecture, 2000. MICRO-33. Proceedings. 33rd An-*

Workload	OrigSizeNoDVFS	OrigSizeWQ0	OrigSizeWQNo	50%IncreasedWQ0	50%IncreasedWQNo
CTC	7107	12361	16060	2980	4183
SDSC	36001	35946	45845	9202	11713
SDSCBlue	4798	6587	8766	2351	3153
LLNLThunder	0	1927	6876	379	1877
LLNLAtlas	69	1841	6691	708	2807

Table 3. Average wait time

nual IEEE/ACM International Symposium on, pages 191–201, 2000.

[2] Y. Ding, K. Malkowski, P. Raghavan, and M. Kandemir. Towards energy efficient scaling of scientific codes. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[3] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 13–23, New York, NY, USA, 2007. ACM.

[4] X. Feng, R. Ge, and K. Cameron. Power and energy profiling of scientific applications on distributed systems. *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 34–34, April 2005.

[5] E. Frachtenberg and U. Schwiegelshohn. New challenges of parallel job scheduling. *Job Scheduling Strategies for Parallel Processing, Springer*, 4942/2008:1–23, April 2008.

[6] V. Freeh, F. Pan, N. Kappiah, D. Lowenthal, and R. Springer. Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 4a–4a, April 2005.

[7] V. W. Freeh and D. K. Lowenthal. Using multiple energy gears in mpi programs on a power-scalable cluster. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 164–173, New York, NY, USA, 2005. ACM.

[8] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal. Analyzing the energy-time trade-off in high-performance computing applications. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):835–848, 2007.

[9] F. Guim and J. Corbalan. A job self-scheduling policy for hpc infrastructures. In *JSSPPS '08: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 51–75. Springer-Verlag, 2008.

[10] J. Hikita, A. Hirano, and H. Nakashima. Saving 200kw and 200 k dollars per year by power-aware job and machine scheduling. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[11] 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. *ipdps*, 0:340, 2006.

[12] C. hsing Hsu and W. chun Feng. A power-aware run-time system for high-performance computing. *sc*, 0:1, 2005.

[13] http://www.cs.huji.ac.il/labs/parallel/workload/. Parallel workload archieve.

[14] S. Kamil, J. Shalf, and E. Strohmaier. Power efficiency in high performance computing. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[15] 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. *sc*, 0:33, 2005.

[16] K. H. Kim, R. Buyya, and J. Kim. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 541–548, May 2007.

[17] B. Lawson and E. Smirni. Power-aware resource allocation in high-end systems via online simulation. In *ICS '05: Proceedings of the 19th annual international conference on Supercomputing*, pages 229–238, New York, NY, USA, 2005. ACM.

[18] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal. Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. *sc*, 0:14, 2006.

[19] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. *SIGPLAN Not.*, 44(3):205–216, 2009.

[20] E. N. (mootaz Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *In Proceedings of the 2nd Workshop on Power-Aware Computing Systems*, pages 179–196, 2002.

[21] A. W. Mu'alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.

[22] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *In Workshop on Compilers and Operating Systems for Low Power*, 2001.

[23] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz. Bounding energy consumption in large-scale mpi programs. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–9, New York, NY, USA, 2007. ACM.