BSLD Threshold Driven Parallel Job Scheduling for Energy Efficient 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

Recently, power awareness in high performance computing (HPC) community has increased significantly. While CPU power reduction of HPC applications using Dynamic Voltage Frequency Scaling (DVFS) has been explored thoroughly, CPU power management for large scale parallel systems at system level has left unexplored. In this paper we propose a power-aware parallel job scheduler assuming DVFS enabled clusters. Traditional parallel job schedulers determine when a job will be run, power aware ones should assign CPU frequency which it will be run at. We have introduced two adjustable thresholds in order to enable fine grain energy performance trade-off control. Since our power reduction approach is policy independent it can be added to any parallel job scheduling policy. Furthermore, we have done an analysis of HPC system dimension. 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. This paper investigates whether having more DVFS enabled processors and same load can lead to better energy efficiency and performance. Five workload logs 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. Applying the same frequency scaling algorithm on 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

Computation performance increase seen last years has been followed by even higher increase in power dissipation and accordingly in energy consumption. Power reduction has become a serious issue even for non-battery operated systems. Power-awareness is present in various systems at different levels. Today, chip manufacturers are faced with power as one of the most important challenges. Since a supercomputer consists of thousands of power hungry microprocessors, it encounters enormously high operating costs and excessive heat that further increases the costs and decreases the reliability. Furthermore, supercomputer operation nowadays results in an excessive environmental footprint due to carbon emission.

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. DVFS technique supports a certain set of frequency-voltage pairs called gears. Running a processor at lower frequency/voltage results in lower power/energy consumption. DVFS is an important OS power management tool in desktop systems. In High Performance Copmuting (HPC) systems, single application savings have been explored applying DVFS to parallel scientific applications [14, 21, 18]. CPU power management of large scale parallel systems at system level has left unexplored. Our aim is to reduce energy consumption of High Performance Computing (HPC) systems running certain jobs at reduced frequency. Best to our knowledge, there is only one attempt to integrate power-awareness into parallel job scheduling but only for scheduling of bag-of-tasks applications ([19]) although power reduction is seen as one of the new challenges of parallel job scheduling [7].

Parallel job scheduler determines when a job will be run. We propose a power-aware parallel job scheduling policy assuming DVFS enabled clusters. Traditional job scheduler determines when a job will be run, while power-aware schedulers should also determine which frequency/voltage it should be run at. In this manner, resource allocation is extended by a power management component. Our work is based on the well established EASY backfilling policy ([23]). Industrial strength schedulers are usually based on backfilling policies ([16, 3]). With backfilling, jobs are not always executed in FCFS order. Users are obliged to submit their estimates of job run times. Certain jobs are allowed to surpass previously submitted jobs if their execution does not delay jobs in the wait queue. Those jobs are backfilled jobs. The EASY backfilling requires only that the first job in the wait queue is not delayed due to backfilled jobs. Altough our work extends the EASY backfilling policy, the frequency scaling algorithm can be applied with any parallel job scheduling policy. As in high performance computing the primary goal is still performance, energy performance trade-off must be done carefully. A scheduler should enable energy performance trade-off control allowing to achieve less energy savings with better overall system performance.

As remarked in [8, 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 energy efficiency can be achieved with no penalty in performance. We have examined a similar scenario but at large system scale. Whether having more processors and running some of them at lower frequencies can result in energy savings while keeping same performance? In this paper we investigate potentials of frequency scaling for enlarged systems regarding both job performance and energy efficiency.

Main contributions of this paper are:

- We propose and evaluate a new power-aware parallel job scheduling policy.

- A mechanism that allows performance-energy trade-off control is designed and analyzed.
- A new simulation infrastructure is developed making it easy to analyze different system and scheduling policy parameters.
- We analyze the impact of system size on energy savings and job performance.

The proposed policy and system size impact on its achievable energy savings are evaluated based on simulation of five workloads of systems in production use. We have upgraded an existing parallel job scheduling simulator to support the BSLD threshold driven scheduling and high level CPU power and execution time models. Our approach decreases CPU energy by 7%- 18% on average depending on allowed job performance penalty. Applying the same frequency scaling algorithm on 20% larger system, CPU energy needed to execute same load can be decreased by almost 30% while having same performance.

The rest of the paper is organized as follows. The next section describes the job scheduling policy we propose. The simulation framework is explained in Section 3. Section 4 describes power and execution time models used by 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

Backfilling job scheduling policies, widely accepted nowadays, improve system performance compared to FCFS policy. They demand that users submit an estimation of job run time. The EASY backfilling algorithm, given in [5], that is executed if the first job in the queue can not start is:

Input:
Queued jobs with requirements
Running jobs with occupied nodes and expected termination
Number of free nodes

Algorithm:

1. Find the shadow time and extra nodes
(a) Sort running jobs according to expected termination time
(b) Find when enough nodes will be available for the first queued job; this is the shadow time
(c) If this job does not need all the available nodes, the ones left over are the extra nodes

2. Backfill job
(a) Loop on the list of queued jobs in order of arrival
(b) For each job check whether either of this conditions is satisfied:
i. The job requires no more than the currently available nodes and will terminate by the shadow time , or
ii. The job requires no more than minimum of currently free nodes and the extra nodes
(d) The first such job can be used for backfilling

If scheduling of a parallel job is seen as determining a rectangle in 2D chart where the x axis presents time and y axis is the number of processors in the system, then backfilling policies schedule small jobs into ”holes” in the 2D chart made by previously arrived jobs that can not be run at the moment.

Widely used metric of user satisfaction is BSLD (Bounded Slowdown). It gives the ratio between the time spent in the system and the job's runtime:

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

where $WaitTime$ and $RunTime$ are time that the job spends waiting on the execution and the execution time. $Threshold$ is a value used to avoid impact of very short jobs on average value. In our experiments it is set to 600 seconds.

The BSLD threshold driven policy runs a job at reduced frequency if at the moment of scheduling its predicted BSLD is less than a previously set $BSLDthreshold$. Predicted BSLD is computed in the following way:

$$PredictedBSLD = max(\frac{WTSched + ReqTime * Coef(f)}{max(Threshold, ReqTime)}, 1). \quad (2)$$

where $WTSched$ is the job's wait time according to the current schedule and $Coef$ is time penalty function that depends on CPU frequency. How frequency scaling affects execution time is described in Section 4. $ReqTime$ is run time estimate that the user submitted.

If run at reduced frequency, a job will be run at the lowest available frequency such that the BSLD threshold condition is satisfied. The scheduler iterates starting from the lowest available CPU frequency trying to schedule a job such that its predicted BSLD is lower than the threshold. If it can not be scheduled at the lowest frequency, the scheduler tries with higher ones.

We have introduced one more parameter, 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). The last threshold is introduced in order to enable fine grain performance-energy trade-off control.

3. Simulation Framework

3.1 The Simulator

Our work is based on simulations as it is usual in the field of parallel job scheduling. We have upgraded the Alvio simulator ([11]) to support DVFS enabled clusters and the proposed scheduling policy. Alvio is an event driven C++ simulator that supports various backfilling policies. While a job scheduling policy determines job run schedule, a resource selection policy determines how job processes are mapped to the processors. We have used First Fit as the resource selection policy in the simulations.

3.2 Workloads

We have used cleaned traces of CTC, SDSC, SDSC-Blue, LLNL-Thunder and LLNL-Atlas workloads from Parallel Workload Archieve [15]. A cleaned trace does not contain flurries of activity by individual users which may not be representative of normal usage. One of the simulator inputs is a workload file in the standard workload format (swf). Each job is represented by single line containing various data fields. We have used the following data fields: job number, submit time, run time, requested time and requested number of processors. Requested time present user estimate of job run time.

The CTC log contains 11 months of accounting records for 512-node IBM SP2 located at the Cornell Theory Center. 430 nodes are dedicated to running batch jobs. The total number of jobs included in the

log is 79 300. 40% of the jobs are sequential jobs and majority of the rest of the jobs require from 2 processors until 64 processors. On the other hand 40% of the jobs have runtime higher than 1 hour and 20% of them have runtime higher than 10 hours. The log presents a workload with many large jobs but with relatively low degree of parallelism.

SDSC and SDSC-Blue logs are from San Diego Supercomputing Center. The SDSC workload, executed on 128 node system, has less sequential jobs than the CTC workload while runtime distribution is very similar. The SDSC-Blue workload has been run on 144 8-way SMP node. There are no sequential jobs in the workload, every jobs asks for at least 8 processors. The log has higher degree of parallelism and shorter jobs comparing to the previous logs.

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. LLNL systems are the biggest simulated systems. They have 4008 and 9216 processors respectively.

Table 1 summarizes workload characteristics. Utilization defined as:

$$SystemUtilization = \frac{\sum_{k=1}^{N_{jobs}} Proc_k * RunTime_k}{N_{proc} * WorkloadTime} \tag{3}$$

is a system performance metric that depends on computational load and on the scheduling policy. Workload utilizations of log parts simulated with the EASY backfilling as the job scheduling policy are given in the table. We have simulated 5000 job part of each workload. Parts of workloads are selected thus not to have many jobs removed.

Workload	Processors	Simulated jobs	Utilization	Average BSLD
CTC	430	20 000 - 24 999	70.09%	4.66
SDSC	128	40 000 - 44 999	85.33%	24.91
SDSC-Blue	1152	20 000 - 24 999	69.17%	5.15
LLNL-Thunder	4008	20 000 - 24 999	79.59%	1
LLNL-Atlas	9216	10 000 - 14 999	75.25%	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:

$$P_{dynamic} = ACfV^2 \tag{4}$$

where A is the activity factor, C is the total capacity, f is the CPU frequency and V is the supply voltage. As we assume the average activity factor of all applications to be same, two different activity factors are used, one for idle CPUs and one 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 values is based on measurements from [6, 17].

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

$$P_{static} = \alpha V \tag{5}$$

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. Static power portion has increased extremely due to permanent technology scaling. 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.

We also compute the energy consumed by a workload when assumed that idle CPUs dissipate no power. Since it is energy needed to execute jobs, we refer to this energy as computational energy. It is introduced to evaluate maximal potentials of the BSLD threshold driven scheduling and system size increase. Although there are trends to have idle processors consuming no power, the other energy model is still more realistic for HPC centers nowadays.

How much frequency scaling affects computation time depends on the application memory boundedness and its communication intensity. The β execution time model is taken from [14]. It is based on the following equation:

$$T(f)/T(f_{max}) = \beta(f_{max}/f - 1) + 1 \tag{6}$$

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. β measures how CPU-bound an application is. $\beta = 1$ means that halving frequency doubles the execution time whereas $\beta = 0$ means that a change in frequency does not affect the execution time. Values of the parameter β for some applications are given in [10] . In our work, β is assumed to be 0.5.

According to the exposed model and the frequency scaling algorithm, total CPU power changes only when a job starts or finishes its execution. The simulator processes START and TERMINATION events and at those moments updates energy consumed by the simulated workload. The energy increment is equal to the product of total CPU power during the interval between the previous START or TERMINATION event and the current one and the interval duration T_k:

$$E_{total} = \sum_{k=1}^{2*N_{jobs}-1} P_k * T_k \tag{7}$$

where N_{jobs} is the total number of jobs and P_k is equal to:

$$P_k = \sum_{i=1}^{n} N_{f_i} * P(f_i) + N_{idle} * P_{idle}. \quad (8)$$

where n is the number of frequencies available in the used DVFS frequency set. $P(f_i)$ is sum of dynamic and static components of CPU power running at the i-th supported frequency. P_{idle} can have two values as we already described. N_{f_i} and N_{idle} are numbers of processors running at the i-th supported frequency and idling, respectively.

5. Results

We report normalized energies consumed by the simulated parts of the workloads. They are normalized with respect to the energies consumed when no frequency scaling is applied. As a measure of performance, we consider average Bounded Slowdown (BSLD) for all simulated jobs. When frequency scaling is applied to a job, its BSLD is computed in the following way:

$$BSLD = max(\frac{WaitTime + PenalizedRunTime}{max(Threshold, RunTime)}, 1) \quad (9)$$

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

5.1 Original System Size

First, we focus on the frequency scaling policy results for the original system sizes. The frequency scaling policy has two parameters $BSLDThreshold$ and WQ_{size}. We have tested three values for $BSLDThreshold$: 1.5, 2 and 3. Four different values are used for WQ_{size} limit. 0 means no DVFS will be applied if there is a job waiting on execution. Less restrictive thresholds are 4 and 16 job limits. The last one puts no limit on the wait queue size. It means that CPU frequency is assigned only based on system utilization.

In Figure 1, energies for two energy scenarios , explained in Section 4, are given. The first graph presents reduction in computational CPU energy when the power-aware scheduler is applied. The second graph gives results assuming that idle processors consume energy. Both graphs give values normalized with respect to the energies without DVFS, assuming that idle processors do not consume or consume energy, respectively.

Applying the BSLD threshold driven scheduling policy, the best energy savings are achieved for CTC, SDSCBlue and LLNLThunder workloads. For the least restrictive combination of parameters ($BSLDThreshold$ = 3 and WQ_{size} = NO LIMIT) savings in computational energy are about 22%. In the case of the most restrictive thresholds savings are 8.5%. The SDSC workload has the worst performance, its average BSLD without frequency scaling is 24.91. Hence the proposed policy with used $BSLDThreshold$ values can not results in energy savings.

Regarding the case when idle processors consume energy, savings are more modest and the energy can be even increased. However, in the least restrictive case ($BSLDThreshold$ = 3 and WQ_{size} = NO LIMIT) relative savings are similar, except for the SDSC workload that because of its original performance does not result in consumed energy decrease.

For a fixed $BSLDThreshold$ increasing the wait queue limit gives frequency schedule that results in lower energy. But for a fixed wait queue limit WQ_{size} an increase in $BSLDThreshold$ does not

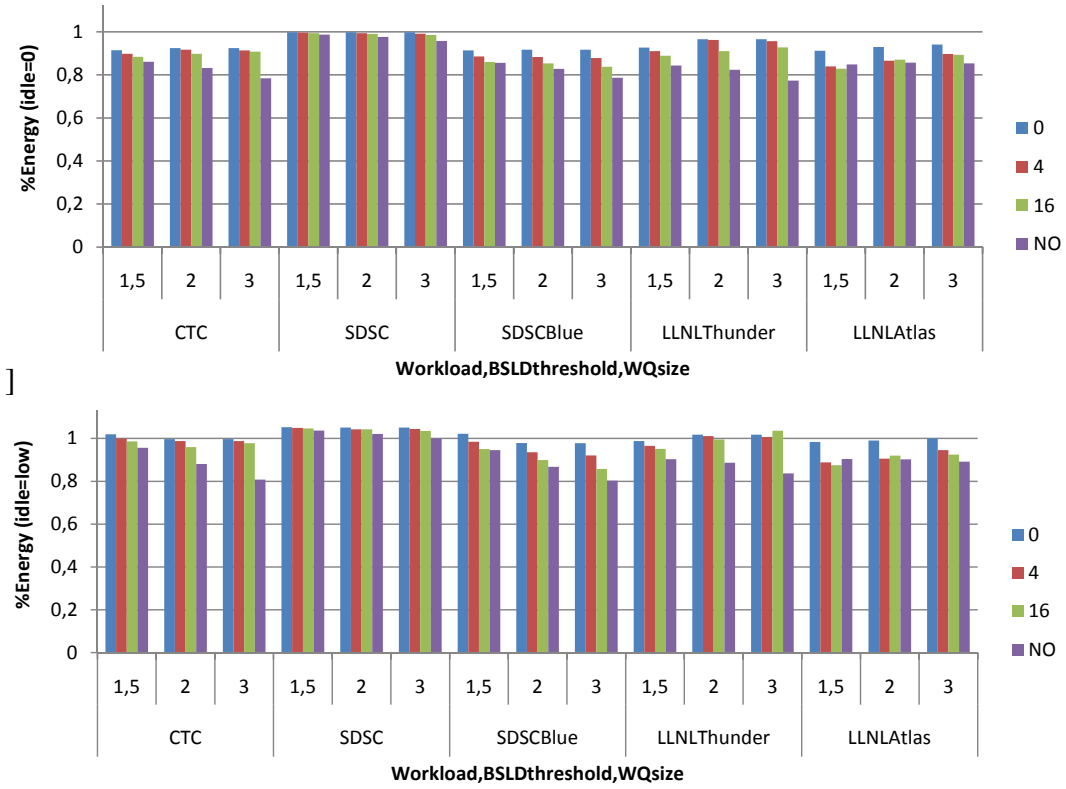

Figure 1. Normalized energy for original system size

necessary save more energy as one might expect. For example, the LLNLThunder workload saves 7.3% of computational energy for $BSLDThreshold$ = 1.5 and WQ_{size} = 0 while for the same WQ_{size} and $BSLDThreshold$ = 2 it saves 3.4% of computational energy. In the first case 835 jobs are run at reduced frequencies while that number in the second case is 560 although the threshold is set to a highest value. Running more jobs at reduced frequencies in the beginning can result in higher BSLD later and less reduced jobs in total. In Figure 2 we can see the number of jobs run at reduced frequency for each workload and parameter combination.

The number of jobs run at reduced frequency has impact on energy savings but more important is their size. For example, for $BSLDThreshold$ = 2 and WQ_{size} = NO LIMIT, the SDSCBlue workload has 2778 jobs run at reduced frequency while for $BSLDThreshold$ = 3 and WQ_{size} = NO LIMIT it has 2654 reduced jobs but their load is higher than in the first case. Hence, in the second case greater savings are achieved.

Average BSLD values are given in Figure 3. As shown in Table 1 average BSLD values varies significantly for different workloads. For example, the SDSC workload (its simulated part) has the highest original average BSLD 24.91. The average BSLD of the LLNLAtlas execution without frequency scaling is 1.08. The best one is LLNLThunder's average BSLD, it is equal to 1. Majority of LLNLThunder jobs are shorter than the threshold from the formula (1), hence their BSLD is 1.

The most aggressive parameter combination $BSLDThreshold$ = 3 and WQ_{size} = NO LIMIT penalizes the most average BSLD but it gives the highest energy savings. Penalty in performance is not always directly proportional to the achieved savings. It depends on the jobs whose frequency has been

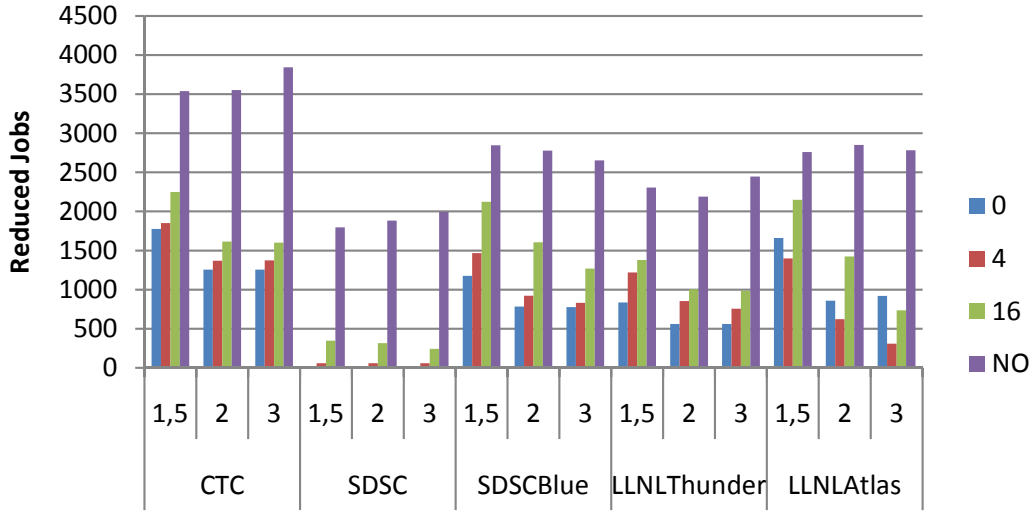

Figure 2. Number of jobs run at reduced frequency

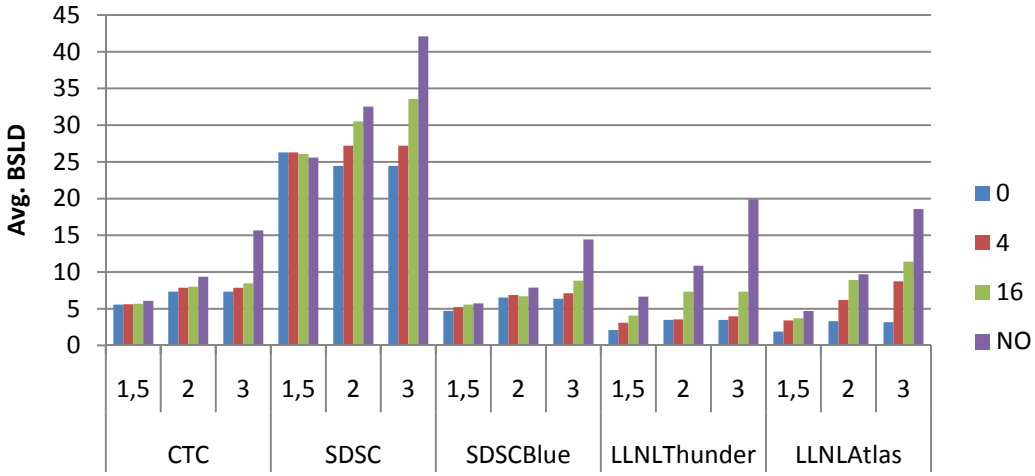

Figure 3. Average BSLD

scaled down. For example, $BSLDThreshold$ = 1.5 and WQ_{size} = 0 threshold combination gives better energy and performance in the case of LLNLAtlas than $BSLDThreshold$ = 2 and WQ_{size} = 0.

5.2 Increased System Size

As already mentioned, different system sizes have been evaluated when the frequency scaling policy has been applied to the five workloads presented in Section 3.2. It has been applied to 10%, 20%, 50% and 75% increased system dimensions. The goal of these experiments is to see how applying DVFS to enlarged systems can reduce operating costs having same or better performance. We have performed a set of experiments with configurations that target the most conservative and the most aggressive scenarios: WQ_{size} = 0 and WQ_{size} = NO LIMIT.

For WQ_{size} = 0 in Figure 4 are given energies normalized with respect to the case of original system without frequency scaling for two scenarios as earlier. WQ_{size} = NO LIMIT energy savings are

presented in Figure 5. We have set $BSLDThreshold$ to the medium used value 2. Logically, computational energy decreases with system dimension increase. Larger system gives more opportunity for frequency reduction as the utilization goes down. Increasing by 20% system size with the BSLD threshold driven policy is possible to decrease computational energy up to 30%. In the other energy scenario, due to the increased number of processors, savings are lower. They are about 15% for CTC, SDSCBlue and LLNLThunder workloads.

In the first energy scenario, increasing system size always results in energy savings. However in the second scenario there is a point for each workload after which further system size increase results in worse energy efficiency due to idle processors.

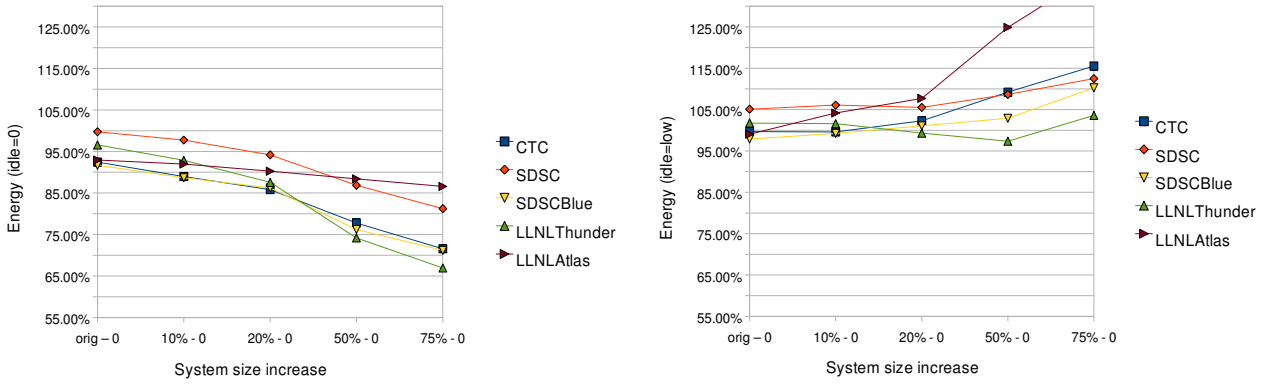

Figure 4. Normalized energies for different system sizes - WQ size = 0

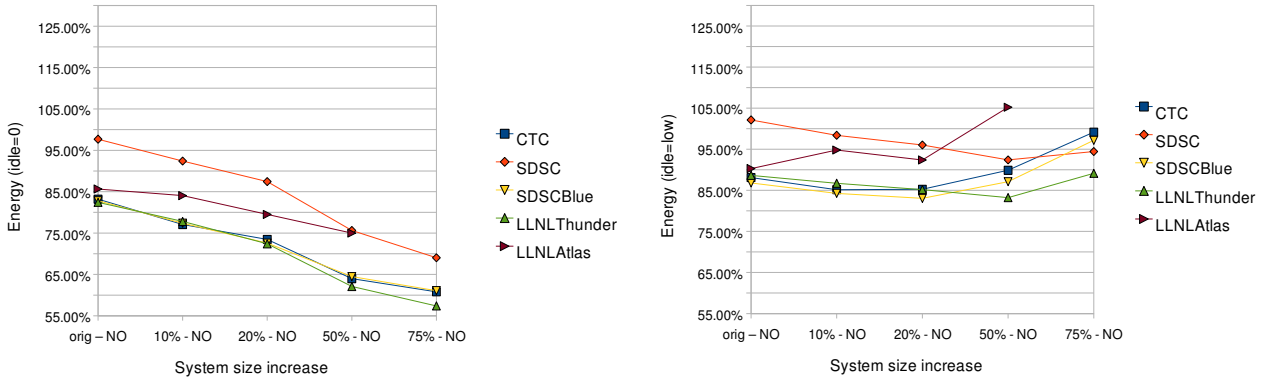

Figure 5. Normalized energies for different system sizes - NO WQ limit

Impact of the BSLD threshold driven policy on enlarged system job performance is shown in Figure 6. We can remark that system size increase always gives better performance although there are more jobs run at reduced frequency. As expected, WQ_{size} = 0 gives better performance results but less energy savings.

It is not possible to improve LLNLThunder and LLNLAtlas performance as their performance without frequency scaling is perfect in BSLD metric. In the way it is defined, frequency scaling always intro-

duces penalty. It can be improved only by reducing job wait time. CTC, SDSC and SDSCBlue improve their average BSLD starting from 10% or 20% system size increase.

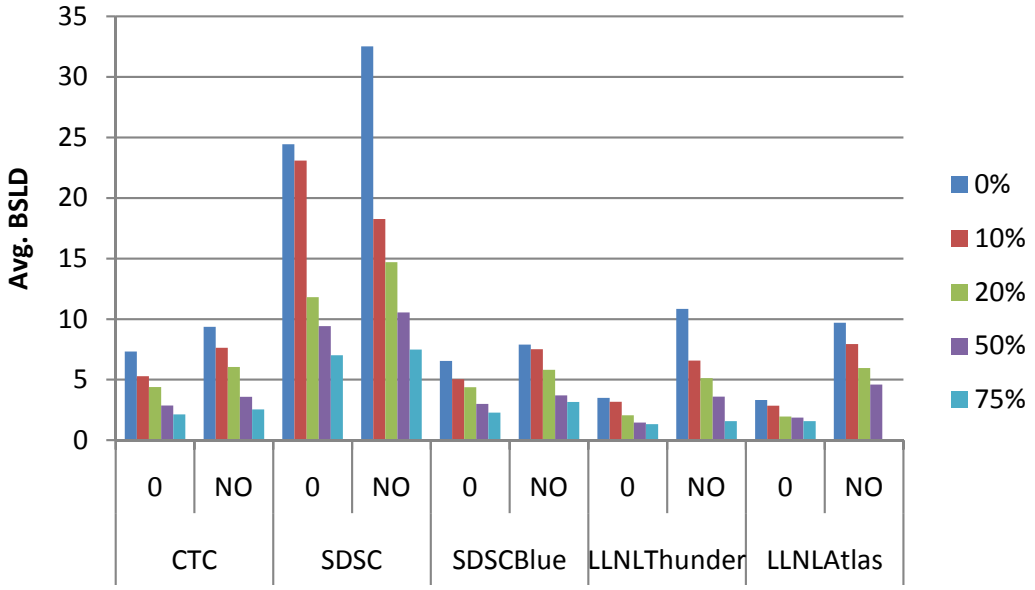

Figure 6. BSLD

6. Related Work

Measurement results of power profiling of parallel applications under different conditions are presented in [8, 17]. Although they just report power and execution time on concrete 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. Rountree et al. developed a system in [25] that determines a bound on the energy saving for an application and an allowable time delay. The system uses a linear programming solver to minimize energy consumed under time constraints. In [9], the idea is to divide programs into phases and then assign different gears to phases according to previous power measurements and a heuristic. In [13] the authors divide program execution into several regions and use an algorithm to select a gear using the execution and power profile. The goal of the algorithm is to minimize the sum of energies or EDP (Energy Delay Product) values per region. Several runtime systems that apply DVFS in order to reduce energy consumed per an application have been implemented. An interval based runtime system that determines at the beginning of each time interval the frequency to use based on an allowable slowdown and an execution time model is developed in[14]. Lim et al., in [21], used an assumption that during communication regions the CPU is not on the critical path. Their runtime system dynamically reduces the CPU frequency during communication regions in MPI programs and it is aimed at communication-intensive codes. In [18], Kappiah et al. developed the Jitter system that exploits load imbalance of MPI applications. All mentioned approaches are concerned with per application power reduction and minimization of frequency scaling penalty on execution time. Usually they can be applied only to certain applications as they are designed to exploit certain application characteristics. Our work targets power management of HPC centers at the level of

whole system that can be applied to all submitted jobs.

Since our work targets supercomputing center power management, work close to ours is presented in [20]. The proposed approach also aims to decrease the power of a supercomputing center, with EASY backfilling used as the job scheduling policy. They reduce the power by powering down some nodes. In this manner job performance metrics 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).The authors in [12] conducted an empirical study on powering down some of system nodes and the decrease in the system power is reported. 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. With new power reduction technologies for all system devices, works based on determining the number of powered on nodes lose their sense.

In [19] Kim et al provide a power aware scheduling algorithm for bag-of-tasks applications with deadline constraints on DVFS enabled clusters. 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 energy efficiency of server clusters. The aggregate power usage characteristics of large collection of servers is given in [4]. The authors also investigated possibility of energy saving using DVFS that has been triggered based on CPU utilization. In [22], 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. Pinheiro et al in [24] also decrease power consumption by turning down cluster nodes under low load. Since shutting a node of their system takes approximately 45 seconds and bringing it back up takes approximately 100 seconds, it was not recommended to simply shut down all unused nodes.

7. Conclusions and Future Work

In this paper we have proposed a power-aware parallel job scheduling policy. It assigns CPU frequency to a job based on the job performance and the number of jobs waiting on execution. The policy has been evaluated for five workload logs from large systems in production used. The evaluation is first performed for original system sizes. Achievable energy savings are inversely proportional to the system utilization. Although we have tested the policy simulating high loaded workloads savings up to 20% in CPU energy are achieved.

Furthermore, we have examined potentials of system size increase regarding both energy efficiency and job performance. It has been concluded that it is possible to reduce energy and improve job performance increasing system size. For example, an increase of 50% in systems size can give much better job performance and up to 35% reduction in computational energy. In order to exploit this idea completely it is necessary to have possibility to turn off idle processors instantaneously. Even if idle processors processors consume energy, increasing system size while applying the proposed frequency scaling algorithm results in better performance while consuming same energy.

As the best combination of the policy parameters regarding energy performance trade-off is workload

dependent, we plan to perform a detail analysis of the parameters in the future work.

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 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 Annual 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] L. R. Dror Feitelson and U. Schwiegelshohn. Parallel job scheduling - a status report. 3277:1–16, 2005.

[4] 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.

[5] D. Feitelson and A. Weil. Utilization and predictability in scheduling the ibm sp2 with backfilling. *Parallel Processing Symposium, International*, 0:0542, 1998.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] D. Jackson, Q. Snell, and M. Clement. Core algorithms of the maui scheduler. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 87–102, 2001.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.