Comput Sci Res Dev (2010) 25: 207–216
DOI 10.1007/s00450-010-0129-x

SPECIAL ISSUE PAPER

Utilization driven power-aware parallel job scheduling

Maja Etinski · Julita Corbalan · Jesus Labarta · Mateo Valero

Published online: 13 August 2010

Abstract In this paper, we propose UPAS (Utilization driven 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 its power dissipation and energy consumption, but introduces a penalty in its performance. Furthermore, performance of other jobs may be affected as their wait times can increase. For this reason, we propose to apply DVFS when system utilization is below a certain threshold, exploiting periods of low system activity. As the increase in run times due to frequency scaling can be seen as an increase in computational load, we have done an analysis of HPC system dimension. This paper investigates whether having more DVFS enabled processors and scheduling jobs with UPAS can lead to lower energy consumption and higher 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 8% on average depending on allowed job performance penalty. Applying UPAS to 20% larger systems, CPU energy needed to execute same workloads can be decreased by 20% while having same or better job performance.

Keywords Power-awareness · Parallel job scheduling

M. Etinski · J. Labarta · M. Valero
Barcelona Supercomputing Center (BSC), Barcelona, Spain

M. Etinski
e-mail: maja.etinski@bsc.es

J. Labarta
e-mail: jesus.labarta@bsc.es

M. Valero
e-mail: mateo.valero@bsc.es

J. Corbalan (✉)
Department of Computer Architecture, Politecnic University of Catalonia (DAC-UPC), Barcelona, Spain
e-mail: juli@ac.upc.edu

1 Introduction

Processor power consumption presents one of the major components of the total system power consumption and it is commonly managed by DVFS (Dynamic Voltage Frequency Scaling) technique. Although DVFS is an important OS power management tool in desktop systems, its use in HPC systems have been explored mainly at the level of single parallel application [11, 16, 19]. Applying frequency scaling having in mind whole workload is still unexplored.

In this paper, we propose Utilization driven Power-Aware job Scheduling policy (UPAS). The main idea of UPAS is to apply DVFS during periods of low system activity. Our proposal relies on previous works that suggest that the goal of achieving near 100% utilization while supporting a real parallel supercomputing workload is unrealistic when jobs submitted to the system are rigid jobs [14]. Analyzing supercomputer workloads from the Parallel Workload Archive [12] it can be found that the most of the workloads have an average system utilization in range of 45%–75%. Furthermore supercomputing systems can have transient periods of low load that occur during night or holidays. Applying DVFS to jobs during periods of low utilization should have a minimum impact on performance since, if the utilization of the system is low, typically there are no jobs waiting for resources. To avoid the situation where the utilization is not very high but there are waiting jobs, we include a second level of policy control. UPAS includes a threshold that

prevents scheduler from running jobs at reduced frequency if there are more jobs in the wait queue than the threshold $WQ_{threshold}$.

UPAS assigns CPU frequency to a job based on system utilization at the job start time. It can be applied with any parallel job scheduling policy. We have used the well established EASY backfilling [22] as the job scheduling policy in our simulations as industrial strength schedulers are usually based on backfilling policies [2, 13]. 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. Jobs executed before previously arrived ones are called backfilled. The EASY backfilling requires only that the first job in the wait queue is not delayed due to backfilled jobs.

This paper makes two key contributions. The first contribution is a proposal and an evaluation of power-aware parallel job scheduling policy. As performance is still the main goal in the HPC community we have explored how to keep performance while reducing energy consumption. It has resulted in the second contribution. The paper explores energy consumption and job performance of enlarged systems with the UPAS scheduling while the number of processors per job remains the same.

We have upgraded an existing parallel job scheduling simulator to support UPAS and high level CPU power and execution time models. Five workloads from systems in production use with up to 9 216 processors are used in simulations. Even for highly loaded workloads UPAS achieves up to 12% of CPU energy reduction. When applied to enlarged systems for same workloads it can improve both energy efficiency and job performance. For example, 20% increase in system size when UPAS is applied can result in 17% of decrease in CPU computational energy and 46% of improvement in job performance.

The power management integrated into job scheduling is exposed in the next section. The simulation framework is explained in Sect. 3. Results are given in Sect. 4 which is followed by related work. Our conclusions are exposed in Sect. 6.

2 HPC job scheduling

In this section we explain the main components of our target environment. Users submit jobs to the scheduler through a file describing job requirements: the number of CPUs required and the requested time . Internally, the HPC scheduler implements the following components:

- *Job scheduling policy*, EASY-Backfilling in this work. The job scheduling policy selects a job to be executed from a list of ready jobs.
- *Resource manager*, it implements the resource selection policy that determines which CPUs will be allocated to a job selected for execution. In this work we implement a First-Fit policy.
- *Frequency assignment policy*, it selects CPU frequency to be set to the allocated processors. Traditionally, HPC schedulers support only two previous components.

2.1 Easy-backfilling

There are many backfilling policies classified by characteristics such as the number of reservations and the prioritization algorithm used in the backfilling queue. When there are less jobs in the wait queue than reservations jobs are executed in FCFS order. If all reservations are used, the algorithm tries to *backfill* jobs from a second queue (the backfilling queue) where jobs are potentially sorted in a different order than by submission time. The number of reservations determines how many jobs in the head of the wait queue will be allocated such that later arrived jobs can not delay them. All time calculations are based on the requested times (user run time estimates). If a job is longer than the user has estimated it, the job will be killed. The EASY-backfilling is one the simplest but still very effective backfilling policy. The backfilling queue is sorted in FCFS order and the number of reservations is set to 1. The EASY-backfilling is executed each time a job is submitted or when a job finishes making additional resources available for jobs in the wait queue.

2.2 Utilization-driven frequency scaling algorithm

The frequency scaling algorithm is interval-based. The interval duration is denoted as T hereafter. A job started during the interval I_j will be run at CPU frequency that depends on the previous interval I_{j-1} utilization. Utilization of the jth time interval, U_j, is equal to:

$$U_j = \frac{\sum_{k=1}^{N_{jobs}} Proc_k * RunTime_k}{N_{proc} * T}, \tag{1}$$

where $Proc_k$ is the number of processors of the kth job that has been executing during the interval I_j and $RunTime_k$ is its execution duration in the interval I_j. N_{proc} is the total number of processors of the system.

CPU frequency assigned to a job depends on two utilization thresholds U_{upper} and U_{lower}. Utilization of previous interval is compared against the thresholds and depending on the results CPU frequency is assigned to the job. Jobs that are submitted to a supercomputing center and wait for execution are in the wait queue. An additional threshold, $WQ_{threshold}$, enables better control over performance loss. It prevents the scheduler from frequency scaling when there are many jobs in the wait queue. If there are more than $WQ_{threshold}$ jobs in the wait queue no DVFS will be applied. A job J_k started during the jth interval runs at the frequency determined according to (2).

$$freq(J_k) = \begin{cases} f_{top} & \text{for } U_{j-1} \geq U_{upper} \text{ or } WQ_{size} > WQ_{threshold}\,, \\ f_{upper} & \text{for } U_{lower} \leq U_{j-1} < U_{upper} \text{ and } WQ_{size} \leq WQ_{threshold}\,, \\ f_{lower} & \text{for } U_{j-1} < U_{lower} \text{ and } WQ_{size} \leq WQ_{threshold}\,. \end{cases} \quad (2)$$

WQ_{size} presents the current number of jobs in the wait queue. Frequencies f_{upper} and f_{lower} depend on the supported DVFS gear set.

UPAS receives from the EASY-Backfilling the job selected to run and determines CPU frequency that will be used for whole job execution. The new requested/run time presents the requested/run time scaled by a factor determined according to the execution time model used in this work (Sect. 3.4). The job scheduler is informed about the new requested time because rescheduling may be needed as before the job's start time its frequency was unknown. The CPU frequency assumed in scheduling before the start time was determined according to (2) at submission time.

At this work, frequency scaling is performed statically, once for whole job execution. As system load normally has no sudden changes, dynamic frequency assignment would result in a different behavior only for very long jobs.

3 Simulation framework

3.1 The simulator

We have upgraded the Alvio simulator [8] to support DVFS enabled clusters and the frequency scaling algorithm. Alvio is an event driven C++ simulator that supports various backfilling policies. The simulator generates job ARRIVAL events based on workload log data. Job START and TERMINATION events are generated according to the job scheduling policy. STATISTICS event is processed after each T interval and system utilization is then computed for the previous interval. Inputs of the simulator are the simulated workload and the power and execution time model parameters. Next sections describe workloads used in the UPAS evaluation and models that determine how frequency scaling affects CPU power consumption and job execution time.

3.2 Workloads

We have used cleaned (recommended) traces of CTC, SDSC, SDSC-Blue, LLNL-Thunder and LLNL-Atlas workloads from Parallel Workload Archive [12]. A cleaned trace does not contain flurries of activity by individual users which may not be representative of normal usage. Thus, the used workload traces do not contain unrepresentative transient phases and they are normally used in job scheduling simulation based works. Workload logs are in the standard workload format (swf). Each job is represented by a 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.

Table 1 summarizes workload characteristics. All the statistics shown here are collected when using the EASY-backfilling policy without UPAS. Columns presented in the table mean, respectively: the name of the workload, the number of processors of the system, average utilization, average load requested (explained later), the number of users, portion of the workload simulated (first-last job), percentage of time with utilization lower than *medium* utilization (80%), and percentage of time with utilization lower than *low* utilization (50%). We have simulated 5000 job part of each workload. Parts of workloads are selected thus not to have many jobs removed.

The column *AvgLR* is the average of the load requested normalized to the system size. If we compare this value with the *AvgUtil* column we can see workloads, such as the LLNLThunder where both values are very similar (79.59% of utilization vs. 0.8 of LR), meaning the system is perfectly dimensioned, and workloads with very high load requested, such as the SDSC, where the average LR is 8.17, showing that the system is clearly overloaded. Rather than the utilization which is a metric that, by its definition, is always in the range (0, 1), the LR gives us an additional information about workloads.

Table 1 Workloads

Workload	Processors	Avg Util	Avg LR	Users	Sim Jobs	%T *medium* U	%T *low* U
CTC	430	70.09%	1.61	679	20K–25K	49.91%	27.67%
SDSC	128	85.33%	8.17	437	40K–45K	26.08%	5.04%
SDSC-Blue	1152	69.17%	2.31	468	20K–25K	54.61%	25.72%
LLNL-Thunder	4008	79.59%	0.8	283	20K–25K	29.47%	10.56%
LLNL-Atlas	9 216	75.25%	0.94	131	10K–15K	25.97%	18.51%

Utilization behavior (*medium* and *low* periods) shows less potential for power reduction with UPAS in cases of SDSC and LLNL-Thunder than in the rest of workloads because the most of the time the system is *high* loaded, specially for SDSC that also has an *AvgLR* very high.

3.3 Power model

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 and static components are equal to:

$$P_{dynamic} = ACfV^2, \quad P_{static} = \alpha V, \tag{3}$$

where A is the activity factor, C is the total capacity, f is the CPU frequency and V is the supply voltage. Hence, dynamic power is proportional to the product of the frequency and the square of the voltage. According to [1] static power is proportional to the voltage. In our model we assume that all applications have same average activity factor. 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 value is based on measurements from [4, 15]. 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. The last row of the table presents normalized average power dissipated per processor for each frequency/voltage pair.

Our power management is performed at high level. As frequency is assigned to a job statically for whole execution overheads due to transitions between different frequencies are negligible. The same applies for entering or exiting low power modes. Nowadays there are low power modes in which power consumption is negligible. Moreover, a design of a whole system that consume negligible power while idling has been proposed [20]. Low power modes give an opportunity to explore full potential of system enlarging and frequency scaling.

Table 2 DVFS gear set

Freq (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
Norm. Power	0.28	0.38	0.49	0.63	0.80	1

3.4 Frequency scaling impact on runtime

In this section we describe the model used to determine new run time at reduced frequency. Usually, due to non-CPU activity (memory accesses and communications) the increase in time is not proportional to the change in frequency. The β model, used in HPC community [11], compares the application slowdown to the CPU slowdown:

$$T(f)/T(f_{max}) = \beta(f_{max}/f - 1) + 1. \tag{4}$$

Based on values of β for different types of applications [7], we have decided to assign β value to each job using the following normal distributions:

- if the number of CPUs is less or equal to 4: $N(0.5, 0.01)$,
- if the number of CPUs is higher than 4 and less or equal to 32: $N(0.4, 0.01)$,
- if the number of CPUs is higher than 32: $N(0.3, 0.0064)$.

In the experiments it is assumed that β values are known at the start time. Otherwise, if the β value is not *a priori* known, the worst case ($\beta = 1$) can be assumed when modeling new requested time at the start time.

4 Evaluation

As it has been already mentioned, we have evaluated UPAS for original and enlarged system sizes (but same workloads). In both cases, we report energy consumption and performance results. Energies are normalized with respect to the energies consumed when no DVFS is applied. As a measure of performance, we consider average Bounded Slowdown (BSLD) of all simulated jobs. BSLD is widely used metric of job performance in high-performance batch scheduling systems. It is defined by

$$BSLD = max\left(\frac{WaitTime + RunTime}{max(RTthreshold, RunTime)}, 1\right), \tag{5}$$

where *WaitTime* presents the time job spent waiting for execution and *RunTime* is its run time. *RTthreshold* is a value used to avoid the impact of very short jobs. In our work *RTthreshold* is set to 10 minutes. When frequency scaling is applied to a job, its BSLD is computed in the following way:

$$BSLD = max\left(\frac{WaitTime + NewRunTime(J, f)}{max(RTthreshold, RunTime)}, 1\right), \tag{6}$$

where $NewRunTime(J, f)$ is the job J run time at the reduced frequency f.

The algorithm parameters are the interval duration T, two utilization thresholds U_{upper} and U_{lower}, and reduced

frequencies f_{upper} and f_{lower}. Analyzing workload utilizations we have decided to set U_{upper} to 80% and U_{lower} to 50%. We have set the higher reduced frequency f_{upper} to 2.0 GHz, the first lower frequency from the used DVFS gear set. The lower reduced frequency f_{lower} is set to 1.4 GHz since among lower frequencies from the DVFS set it has the best ratio between energy reduction and penalty in execution time. Two values for the parameter T, 10 minutes and 1 hour, are used in the simulations. As the difference in results is 1% or less, the algorithm is not very interval duration sensitive. 10 min interval shows slightly better results in both energy and performance. Hence, in the paper we give results for 10 minute interval.

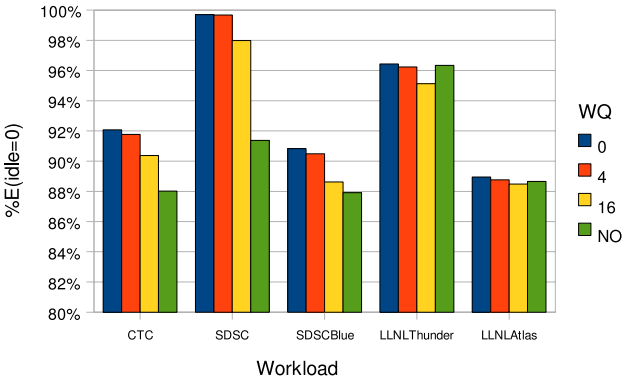
4.1 UPAS evaluation: original system size

In Fig. 1 energies for two energy scenarios, that differ in power consumption of idle CPUs, are given. The upper graph presents reduction in computational CPU energy (energy needed to execute jobs) when the power-aware scheduler is applied. The lower graph gives results assuming that idle processors consume power which is equal to idle power at the lowest available frequency (21% of top running power in our case). Both graphs give values normalized with respect to the energies without DVFS, assuming that idle processors do not consume or consume energy, respectively. Four different values of $WQ_{threshold}$ are selected for evaluation. 0 means no DVFS will be applied if there is any job waiting. Less restrictive thresholds are 4 and 16. The last one puts no limit on the wait queue size (frequency is assigned only based on system utilization).

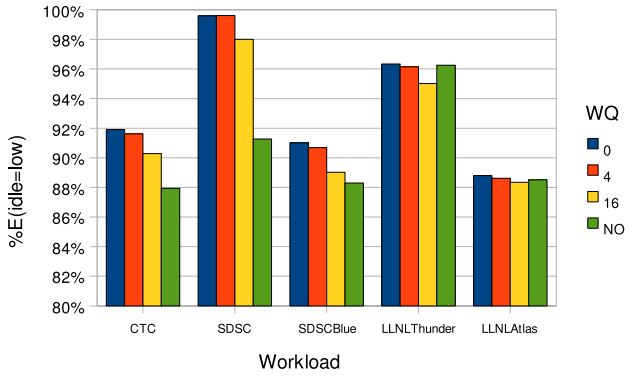
The highest energy savings are achieved for LLNLAtlas, CTC and SDSCBlue workloads, 12% in the most aggressive case ($WQ_{threshold}$ = no limit). Per workload system utilizations are given in Table 1. Since SDSC and LLNLThunder have higher system utilization UPAS has less opportunity to scale down CPU frequencies. Hence their savings are more modest, 4% and 8% in the best case. Relative energy savings of two energy scenarios are almost equal.

Regarding the impact of the $WQ_{threshold}$ parameter, the highest energy savings are achieved when no limit is imposed. Differences are 4%–8% between using the limit of 0 jobs and no limit. However for bigger systems, LLNLThunder and LLNLAtlas, there is almost no difference for 4, 16 and no limit thresholds implying that their wait queues are usually shorter. Hence $WQ_{threshold}$ needs to be adjusted taking into account the workload.

Average BSLD values are given in Fig. 2. We present values normalized to the average workload BSLD of executions without UPAS. Original average BSLD values vary significantly for different workloads. For example, the SDSC workload has the highest original average BSLD 24.91. The average BSLD of the LLNLAtlas execution without UPAS



(a) $P_{idle} = 0$



(b) $P_{idle} > 0$

Fig. 1 Normalized energies for original system size

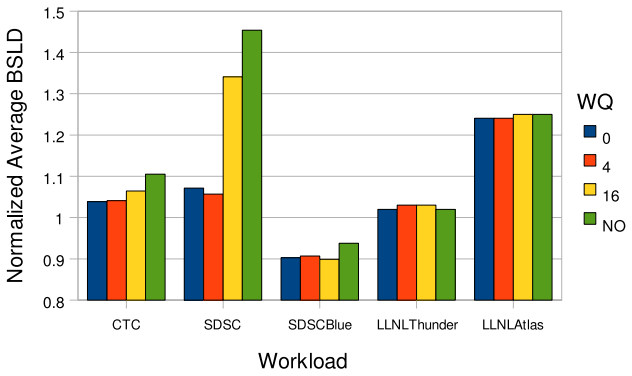


Fig. 2 Performance results for original system size

is 1.08. The best one is LLNLThunder's average BSLD, it is equal to 1. Relative penalty in performance is not always proportional to achieved savings. Although less restrictive frequency scaling per workload (higher the $WQ_{threshold}$ parameter) results in an increase in the average BSLD value, similar energy savings for different workloads can result in different relative performance penalty. The increase in the average BSLD of a workload comes from two reasons. According to how we defined BSLD for jobs executed at re-

duced frequency by (6), scaling a job frequency down increases its BSLD as it increases the job execution time. Moreover, as in this way system load is artificially increased, certain jobs might wait longer for execution. Longer wait time means higher BSLD and that is the second reason for the performance decrease. LLNLAtlas and CTC achieve similar relative energy savings with UPAS but LLNLAtlas suffers from higher relative performance penalty. LLNLAtlas jobs wait on average much shorter on execution than CTC jobs. Hence considering how BSLD metric is defined (5), an increase in execution time of jobs run at reduced frequency affects more the overall performance in the case of LLNLAtlas. The SDSCBlue workload shows an improvement in both performance and energy. As frequency scaling changes job run times, with backfilling it can give a different schedule. There are more backfilled jobs when UPAS is applied to the SDSCBlue workload. Their BSLD improvement gives an improvement in the overall performance.

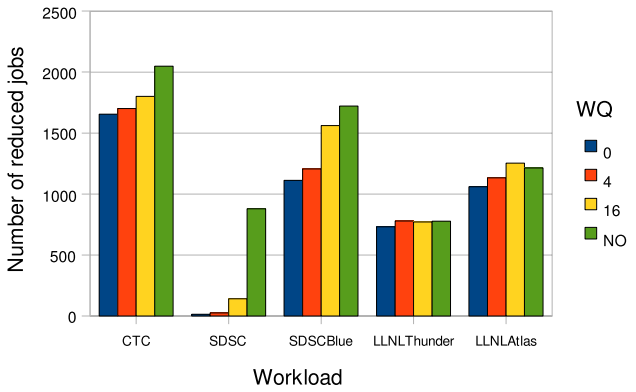


Fig. 3 Number of jobs executed at lower frequency

The number of jobs that have been run at reduced frequency for different values of $WQ_{threshold}$ is shown Fig. 3.

The first graphic of Fig. 4 presents SDSCBlue's utilization behavior during time. The second one shows how average frequency of running CPUs follows changes in utilization while the last graphic gives the number of jobs in the wait queue.

UPAS has been designed as a conservative approach to reduce energy consumed by HPC centers. Hence penalty in job performance is not very high. On the other hand, energy savings are modest. We have investigated system size enlarging that gives UPAS more opportunity for frequency scaling due to lower utilization. Furthermore, system enlarging amortizes performance penalty due to a decrease in wait time.

4.2 UPAS evaluation: system size evaluation

UPAS has been applied to 10%, 20%, 50% and 75% increased system dimensions. We have performed a set of experiments with configurations that target the most conservative and the most aggressive scenarios: $WQ_{threshold} = 0$ and $WQ_{threshold} =$ no limit. For $WQ_{threshold} = 0$ in Fig. 5 are given energies normalized with respect to the case of original system without UPAS for two scenarios as earlier. $WQ_{threshold} =$ no limit energy savings are presented in Fig. 6. Logically, computational energy decreases with system dimension increase. Larger system gives more opportunity for frequency reduction as utilization goes down. Regarding computational energy, slopes of different workloads are very similar. Generally, first 10% of system size contribute in relative energy saving as much as 25% of increase starting from a 50% enlarged system. In the second scenario

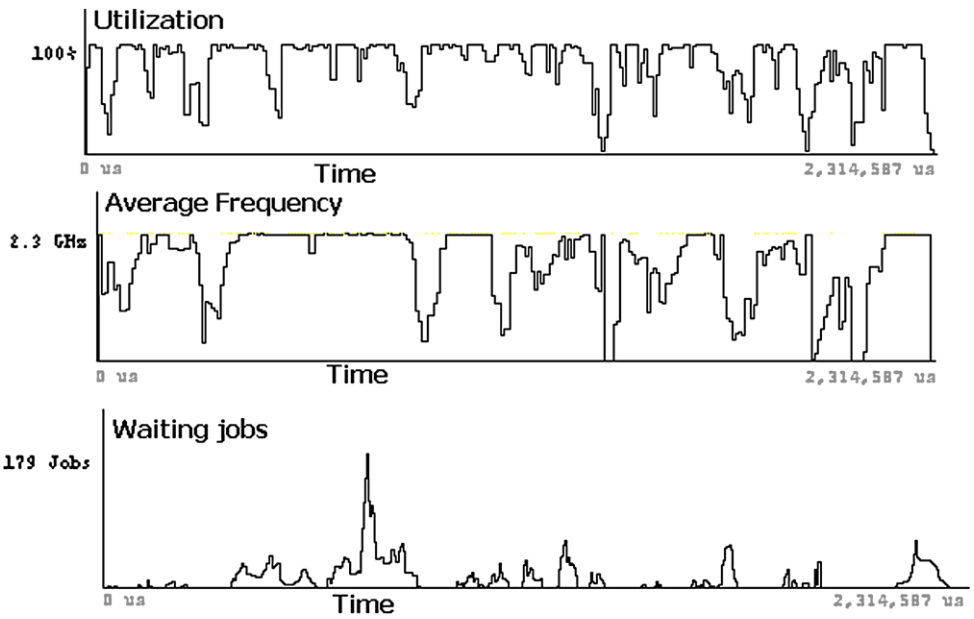
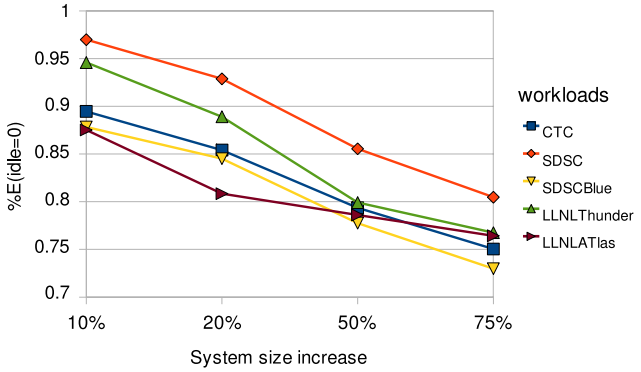
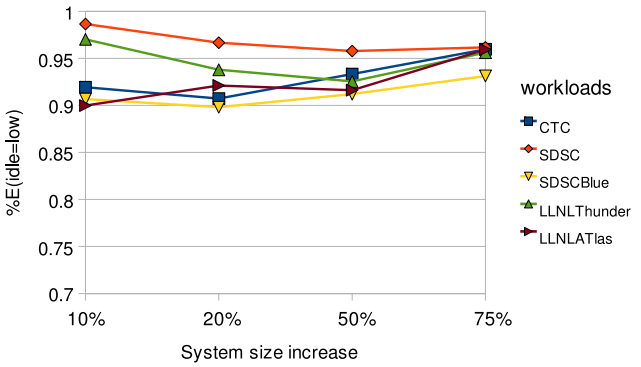


Fig. 4 SDSCBlue—utilization, average frequency and waiting jobs

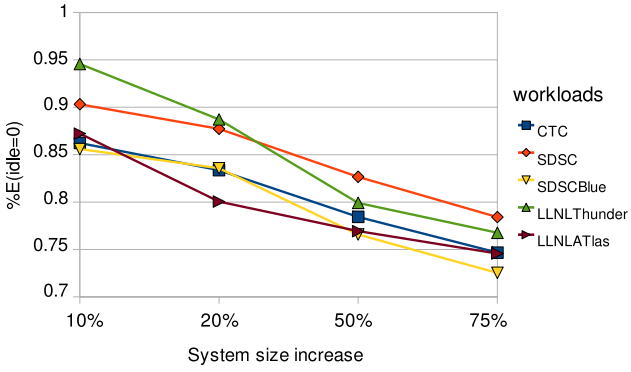


(a) $WQ_{threshold} = 0$ and $P_{idle} = 0$

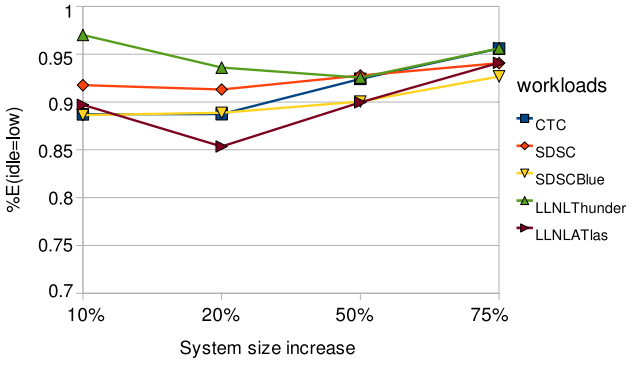


(b) $WQ_{threshold} = 0$ and $P_{idle} > 0$

Fig. 5 Normalized energies for different system sizes—WQ size = 0



(a) $WQ_{threshold} = NO$ and $P_{idle} = 0$



(b) $WQ_{threshold} = NO$ and $P_{idle} > 0$

Fig. 6 Normalized energies for different system sizes—NO WQ limit

when idle processors dissipate power there is a point of the minimal energy consumption. An increase in system size over this point while applying UPAS will not give better energy. Nevertheless even for 75% system size increase, the energies with UPAS are lower than the original ones in all cases except one, LLNLAtlas for $WQ_{threshold} = 0$.

$WQ_{threshold} = 0$ and $WQ_{threshold} =$ no limit differ less for enlarged systems than for the original ones as wait queues contain less jobs due to more processors available.

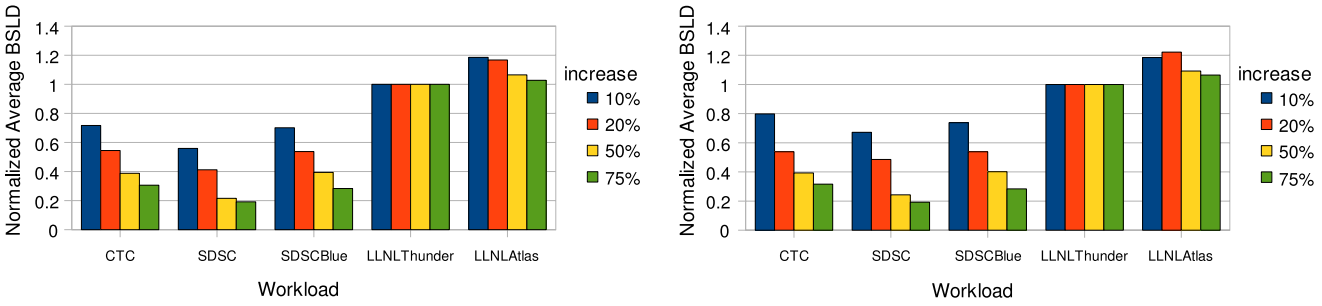
Increasing system size by 50% and applying the power-aware scheduler can reduce computational energy by 23.5%. Even more loaded workloads, SDSC and LLNLThunder achieve savings of 20%. 20% system size increase shows better ratio between system size increase and energy reduction.

Improvement in job performance is shown in Figs. 7(a) and 7(b) for $WQ_{threshold} = 0$ and $WQ_{threshold} =$ no limit respectively. Increasing system size while applying the power-aware scheduler reduces the average BSLD significantly. Performance of CTC, SDSC and SDSCBlue workloads is much better. For example, an increase of 20% in system size gives approximately 50% of improvement in performance in spite of frequency scaling. As LLNLThunder and LLNLAtlas have already perfect performance in the used metric, it was not possible to improve it. Due to its short jobs, the LLNLThunder average BSLD stayed stable. There is some penalty in LLNLAtlas performance due to the increase in job execution times.

5 Related work

There are works done at level of one application. Power profiling of parallel applications is done in some works [5, 15]. 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 [6, 10, 24]. Several runtime systems that apply DVFS in order to reduce energy consumed per application are implemented [11, 16, 19]. 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 presents system level power management of large scale systems. Lawson et al. aim to decrease supercomputing center power dissipation powering down some nodes [18]. The EASY backfilling is used as the job scheduling policy. In this manner BSLD is affected



(a) $WQ_{threshold}$=0 (b) $WQ_{threshold}$=NO

Fig. 7 BSLD for increased size systems

seriously in cases of high load. Two policies are proposed to determine the number of active nodes of the system. The first policy 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 [9]. 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 [17]. 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. 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 [21]. Pinheiro et al. also decrease power consumption by turning down cluster nodes under low load [23]. 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.

6 Conclusions

In this paper, we have proposed UPAS, a power-aware parallel job scheduler. It has been evaluated for five workloads with up to 9 216 processors. Applying UPAS to the original systems results in CPU energy savings starting from 4% for a very loaded system to 12% for less loaded systems. Workloads used in the simulations have high utilizations. As it is shown in Sect. 4, achievable savings are directly proportional to system utilizations. Hence, for less loaded systems one should expect higher savings. Higher threshold allows for more savings but results in worse job performance. Increase in average BSLD is manageable via $WQ_{threshold}$.

As new technologies allowing use of very low power modes for idle hardware have emerged, we have evaluated UPAS on enlarged systems considering energy consumption and job performance. Increasing system size by 50% and applying the power-aware scheduling can reduce computational energy by 23.5% and give significant performance improvements. 20% system size increase shows better ratio between system size increase and energy reduction. In future, determination of system size should be guided not only by desired job performance but by power-awareness as well. Investing more in equipment might result in later savings in operating costs as nowadays the total cost of ownership starts to be dominated by operating costs. In order to get precise estimation of possible benefit of system enlarging one should consider all system components and their power reduction techniques. That is out of the scope of this work but may present interesting idea for future work.

Acknowledgements 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. Butts J, Sohi G (2000) A static power model for architects. In: Proceedings of the 33rd annual IEEE/ACM international symposium on microarchitecture, MICRO-33, 2000
2. Dror Feitelson LR, Schwiegelshohn U (2005) Parallel job scheduling—A status report

3. Fan X, Weber W-D, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: ISCA '07: Proceedings of the 34th annual international symposium on computer architecture
4. Feng X, Ge R, Cameron K (2005) Power and energy profiling of scientific applications on distributed systems. In: Proceedings of the 19th IEEE international symposium on parallel and distributed processing, 2005
5. Freeh V, Pan F, Kappiah N, Lowenthal D, Springer R (2005) Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster. In: Proceedings of the 19th IEEE international symposium on parallel and distributed processing, 2005
6. Freeh VW, Lowenthal DK (2000) Using multiple energy gears in MPI programs on a power-scalable cluster. In: PPoPP '05: Proceedings of the 10th ACM, SIGPLAN symposium on principles and practice of parallel programming
7. Freeh VW, Lowenthal DK, Pan F, Kappiah N, Springer R, Rountree BL, Femal ME (2007) Analyzing the energy-time trade-off in high-performance computing applications. IEEE Trans Parallel Distrib Syst
8. Guim F, Corbalan J (2008) A job self-scheduling policy for HPC infrastructures. In: JSSPPS '08: Proceedings of the workshop on job scheduling strategies for parallel processing
9. Hikita J, Hirano A, Nakashima H (2008) Saving 200 kw and 200 k dollars per year by power-aware job and machine scheduling. In: IEEE international symposium on parallel and distributed processing, IPDPS 2008
10. Hotta Y, Sato M, Kimura H, Matsuoka S, Boku T, Takahashi D (2006) Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster. In: IPDPS, 2006
11. Hsu CH, Feng WC (2005) A power-aware run-time system for high-performance computing. In: SC, 2005
12. http://www.cs.huji.ac.il/labs/parallel/workload/. Parallel workload archive
13. Jackson D, Snell Q, Clement M (2001) Core algorithms of the Maui scheduler. In: Proceedings of the workshop on job scheduling strategies for parallel processing, 2001
14. Jones JP, Nitzberg B (1999) Scheduling for parallel supercomputing: a historical perspective of achievable utilization. In: IPPS/SPDP '99/JSSPP '99: Proceedings of the job scheduling strategies for parallel processing
15. Kamil S, Shalf J, Strohmaier E (2008) Power efficiency in high performance computing. In: IEEE international symposium on parallel and distributed processing, IPDPS 2008
16. Kappiah N, Freeh VW, Lowenthal DK (2005) Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs. In: SC, 2005
17. Kim KH, Buyya R, Kim J (2007) Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In: CCGRID, 2007
18. Lawson B, Smirni E (2005) Power-aware resource allocation in high-end systems via online simulation. In: ICS '05: Proceedings of the 19th annual international conference on supercomputing
19. Lim MY, Freeh VW, Lowenthal DK (2006) Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs. In: SC, 2006
20. Meisner D, Gold BT, Wenisch TF (2009) Powernap: eliminating server idle power. In: SIGPLAN Not, 2009
21. (Mootaz) Elnozahy EN, Kistler M, Rajamony R (2002) Energy-efficient server clusters. In: Proceedings of the 2nd workshop on power-aware computing systems, 2002
22. Mu'alem AW, Feitelson DG (2001) Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Trans Parallel Distrib Syst
23. Pinheiro E, Bianchini R, Carrera EV, Heath T (2001) Load balancing and unbalancing for power and performance in cluster-based systems. In: Workshop on compilers and operating systems for low power
24. Rountree B, Lowenthal DK, Funk S, Freeh VW, de Supinski BR, Schulz M (2007) Bounding energy consumption in large-scale MPI programs. In: SC '07: Proceedings of the 2007 ACM/IEEE conference on supercomputing

Maja Etinski is a PhD student at Technical University of Catalonia (UPC) and the Barcelona Supercomputing Center (BSC-CNS). She graduated at Department of Mathematics and Computer Science at Belgrade University in 2007 and finished her mater studies in 2009 at UPC. Her PhD research is in the field of Power-aware job scheduling for high-performance computing centers.

Julita Corbalan is an associate professor at the Computer Architecture Department at UPC, in Barcelona and an associate researcher of the Barcelona Supercomputing Center (BSC-CNS). Since 2001 she has been lecturing mainly on Operating System courses. Her research interests includes processor management of openmp, mpi, and mpi + openmp applications, openmp runtime management and power-aware high performance computing. She has participated in several long-term research projects with other universities and industries, mostly in the framework of the European Union ESPRIT and IST programs.

Jesus Labarta is full professor on Computer Architecture UPC since 1990. Since 1981 he has been lecturing on computer architecture, operating systems, computer networks and performance evaluation. His research interest has been centered on parallel computing, covering areas from multiprocessor architecture, memory hierarchy, parallelizing compilers, operating systems, parallelization of numerical kernels, GRID and performance analysis and prediction tools. He has leaded the technical work of UPC in 15 industrial R + D projects. Significant performance improvements were achieved in commercial codes owned by partners with whom he has cooperated. From 1995 to 2005 he was director of CEPBA where he has been highly motivated by the promotion of parallel computing into industrial practice, and especially within SMEs. In this line he has acted as responsible of three technology transfer cluster projects where his team managed 28 subprojects. His major directions of current work relate to performance analysis tools and OpenMP. From 2000 to 2004 he is strongly committed in carrying out and promoting productive research cooperation with IBM as part of the CEPBA-IBM Research Institute. He is now involved in the Barcelona Supercomputing Center leading the research on high performance computing.

Mateo Valero has been a full professor in the Computer Architecture Department, Universitat Politècnica de Catalunya (UPC), since 1983. Since May 2004, he has been the director of the Barcelona Supercomputing Center (the National Center of Supercomputing in Spain). His research topics are centered in the area of high-performance computer architectures. He has coauthored more than 500 publications. He has served in the organization of more than 300 international conferences. His research has been recognized with several awards, including two Spanish National Awards on Informatics and on Engineering, the Rey Jaime I Award in basic research, and the prestigious Eckert-Mauchly Award. He received a Favourite Son Award from his hometown, Alfamén (Zaragoza), which named their public college after him. He is a fellow of the IEEE and the ACM. He is an academic of the Royal Spanish Academy of Engineering, a correspondant academic of the Royal Spanish Academy of Sciences, an academic of the Royal Academy of Science and Arts and an academic of the European Academy, and a doctor honoris causa at the universities of Chalmers, Belgrade, Las Palmas de Gran Canaria in Spain and Veracruz in México.