Power-Aware Load Balancing Of Large Scale MPI Applications

Maja Etinski[†]
maja.etinski@bsc.es

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

Jesus Labarta[†]
jesus.labarta@bsc.es

Mateo Valero[†]
mateo.valero@bsc.es

Alex Veidenbaum[‡]
alexv@ics.uci.edu

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

[‡]Department of Computer Science
University of California, Irvine CA

Abstract

Power consumption is a very important issue for HPC community, both at the level of one application or at the level of whole workload. Load imbalance of a MPI application can be exploited to save CPU energy without penalizing the execution time. An application is load imbalanced when some nodes are assigned more computation than others. The nodes with less computation can be run at lower frequency since otherwise they have to wait for the nodes with more computation blocked in MPI calls. A technique that can be used to reduce the speed is Dynamic Voltage Frequency Scaling (DVFS). Dynamic power dissipation is proportional to the product of the frequency and the square of the supply voltage, while static power is proportional to the supply voltage. Thus decreasing voltage and/or frequency results in power reduction. Furthermore, over-clocking can be applied in some CPUs to reduce overall execution time. This paper investigates the impact of using different gear sets , over-clocking, and application and platform propreties to reduce CPU power. A new algorithm applying DVFS and CPU over-clocking is proposed that reduces execution time while achieving power savings comparable to prior work. The results show that it is possible to save up to 60% of CPU energy in applications with high load imbalance. Our results show that six gear sets achieve, on average, results close to the continuous frequency set that has been used as a baseline.

1. Introduction

HPC systems use an increasingly large number of processors with MPI-based parallel programs. The resulting increase in the number of processes of an application usually decreases the load balance degree of the application. Several solutions have been proposed to load balance applications via processor resource distribution ([1]). Via Dynamic Voltage Frequency Scaling (DVFS) that enables per processor frequency control it is possible to balance imbalanced applications in power aware manner. In load imbalanced MPI applications, there are processes which complete their computation and have to wait for the other processes to communicate. These nodes can run at lower frequencies and save energy consumed by CPU without increasing the execution time. The dynamic power is proportional to the product of the frequency and the square of the voltage and it can be reduced via DVFS technique. A processor that supports DVFS defines a set of gears i.e. frequency-voltage pairs at which it can run. The idea to exploit load imbalance to save energy has been already proposed and a runtime system called Jitter has been implemented and described in [18]. A static version of this approach is called *Slack* and it is observed in [21].

Prior work has been evaluated only on small clusters, for instance eight nodes were used in ([21], [18]). Larger systems are likely to have a more unbalanced execution. In this work we investigate the impact of the cluster size on imbalance and show that it can grow significantly with cluster size. Thus larger scale applications may have a greater load imbalance and therefore allow greater relative savings than the small clusters. This paper presents results for large clusters with up to 128 nodes and for real world applications. Furthermore, we show the correlation between the degree of load balance and the energy reduction.

The Jitter system proposed an algorithm which computes for each process the frequency necessary to complete the computation at the same time as other processes, based on slack time. We will refer to this algorithm as the MAX algorithm, because it makes all processes complete in the time of the longest running process. Furthermore, this name makes clear the difference between this already proposed approach

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

and an algorithm called AVG that we propose. The main difference between the MAX and average algorithms is that the AVG algorithm tries to reduce the maximum time by using over-clocking. Our conjecture was that over-clocking would need to be applied only in very few processors and thus not lead to an increase in the power consumption.

Several issues have a major effect on DVFS load balancing of large scale MPI applications and are investigated in this paper. First is the question of how to manage DVFS, discrete versus continuous frequency choice, the number of discrete setting, etc. We investigated which is the most appropriate DVFS gear set size and how frequencies should be distributed to save more energy. Second, hardware dependent parameters, such as the fraction of static power in the total CPU power and the CPU activity factor are investigated. Third is the question of how memory bound is an application. This can have a significant effect on the execution time. Memory boundedness determines energy-time tradeoff for sequential and load balanced parallel applications when whole applications are run at reduced speed, as was observed in [7] but it is especially important in load imbalanced applications when different CPUs run at different frequencies.

This paper explores the use of DVFS techniques that may not have been implemented in hardware, for instance different gear sets. Thus simulation is used to evaluate execution time and power savings. A new simulation infrastructure was created to support this research. The following simulation approach is used. First, we collect traces of observed applications. Next, they are processed to get per MPI process data. Based on this information and according to the algorithm used, the per CPU frequencies are assigned so that computation times per process are balanced. One frequency is assigned to a process for the whole execution. This approach can be applied to any application with regular, iterative behavior. Finally, the execution time is simulated based on a model described in 3.2 and the original and the new CPU energies are compared.

The results presented in this paper show that large scale applications can potentially save a significant amount of CPU energy, up to 60%, in imbalanced MPI applications. With respect DVFS gears, our results for the set of applications studied show that six DVFS gears are sufficient to achieve gains that are close to the ideal case of a continuous frequency set.

Main contributions of this paper are:

- A new algorithm AVG that reduces both the energy consumption and the execution time via DVFS and overclocking. Power dissipation and execution time using the new algorithm and using the MAX algorithm are presented for up to 128 processors
- The impact of CPU and application characteristics on energy and execution time is investigated
- A new simulation methodology and infrastructure are proposed making it easy to analyze different applications, algorithms, and system parameters
- To the best of our knowledge, this is the first study of load imbalance driven energy and execution time reduction for large clusters, with up to 128, and for real world applications
- It is shown that continuous frequency scaling is not necessary and a small set of voltage-frequency gears can achieve almost identical results.

The rest of the paper is organized as follows. Section 2 exposes related work. Section 3 describes load balancing algorithms, the models of power and the execution time and tested DVFS gear sets. The simulation infrastructure is described in section 4. Metrics used to compare results, description of the observed applications and results are given in section 5. The last section contains conclusions and future work.

2. Related Work

Prior work on power management addressed CPU, memory, disk interconnection network, and whole-system power. In this section we focus on CPU power reduction in parallel applications via DVFS since it is the work most closely related to ours.

There are many works from the field of power profiling of applications. Although they just report measurement results of power and execution time on concrete platforms for different gears or numbers of nodes, they give a valuable insight in relations between frequency, power and execution time. In [20] Pan et al. investigate the tradeoff between energy and time when applications run at lower frequencies. They tested sequential HPC programs on two different platforms. Freeh et al. explore energy consumption and execution time of parallel applications across different numbers of nodes in [5] . All measurements are done using single gear for the whole execution. They found that for some programs it is possible to both consume less energy and execute in less time when using a larger number of nodes, each running at reduced speed. In [17], power measurements are done for different applications per node, cabinet or the whole system of almost 20,000 nodes. Power profiles of NAS parallel benchmarks per component (CPU, memory, disk and NIC) and by system scale are provided in [4].

Also, there are works based on previous profiling of the application. Rountree et al. developed a system in [21] 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 [6], the idea is to divide programs into phases and then assign different gears to phases according to previous measurements and a heuristic. In [9] the authors divide the 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 values per region. Afterwards, applications are run at the selected combination of gears and the energy is measured. Ge et al. investigated three different strategies how to apply DVFS to decrease CPU power in [8].

Several runtime systems that apply DVFS in order to save energy have been implemented. Interval based runtime system determine at the beginning of each time interval the frequency to use based on an allowable slowdown and an execution time model [10]. The time model was based on the MIPS rate and its goal was to correlate the execution time with CPU frequency changes. Lim et al., in [19], 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. As we already mentioned, the MAX algorithms is a static version of this approach.

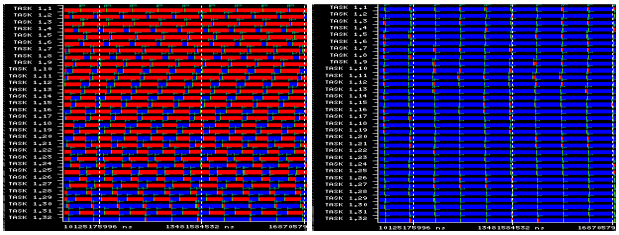
3. Power Aware Load Balancing

3.1 Algorithms

In a MPI application there are processes that finish computation phase before others and then are blocked waiting for communication. According to the MAX algorithm, the frequency of a CPU is determined so that the computation time of a process that runs on the CPU would be equal to the original maximal computation time among all processes. Thus after frequency scaling all processes complete the computation phase at approximately the same time. This basically assumes an iterative application behavior with fixed computation time ratio among processes so that the frequencies can be set statically. In this way, the CPU that has the highest load runs at the top frequency and other CPUs run at reduced frequencies. How much a CPU frequency should be decreased depends on the amount of the computation load that is assigned to the process but also on memory boundedness of the code. The execution time model taking these into account is described in the next section.

Since it is not possible to scale frequency to an arbitrary value, except in the ideal case of continuous frequency set, the new frequency is the closest higher frequency from the gear set than the frequency that should be assigned according to the algorithm.

In Figure 1 a visualization of a part of BT-MZ execution is given before and after the MAX algorithm is applied assuming continuous frequency scaling. In the original execution a lot of time was spent waiting for communication while under the MAX algorithm almost all the time is spent in computation.



(a) Original execution (b) After MAX algorithm

Figure 1. Visualization of BT-MZ execution

The algorithm AVG aims to balance the computation times to the average, rather than maximum, original computation time. This would require a frequency increase in one or more CPUs. When an application is very load imbalanced, it may not be possible to scale a high load process to the average value because it would need an unrealistically high frequency. In this work we use frequency thresholds of 10 and 20% above the manufacturer's specified maximum frequency (e.g. amount of over-clocking). Whenever because of high degree of load imbalance is not possible to scale all computation times to the average value, the frequencies are determined so that the target computation time is the closest one to the average but attainable with the available frequency range. In [22], dynamic processor overclocking was proposed to improve performance under a power constraint. This paper investigates how DVFS load balancing can benefit from processor over-clocking and whether it is possible to reduce both the energy and the execution time.

3.2 Power and Time Models

CPU power represents a significant fraction of the overall system power. The exact percentage is system dependant, but the CPU power can make approximately 45-55% of overall system power ([18]) . Hence, it is important to decrease the CPU power in order to save power. The CPU power consists of the static and the dynamic power. The dynamic power is equal to:

$$P_{dynamic} = ACfV^2 \quad (1)$$

where A is the activity factor, C is the total capacity f is the CPU frequency and V is the supply voltage. The ratio between computation and communication activity factors depends on the program and the concrete processor. In

this study we initially assume it to be 1.5 and later investigate other values. These values are based on measurements from [4, 17].

The static power is equal to ([2]):

$$P_{static} = \alpha V \quad (2)$$

where α presents a technology and design dependent parameter. We assumed in our baseline case that static power equals 20% of total CPU power when the CPU is loaded and runs at the top frequency as it is assumed in [3]. It is used to determine parameter α. Since the fraction of static power in total CPU power increases as technology scales down, higher percentages of static power are also investigated.

While the duration of communication does not depend on the CPU frequency, duration of computation depends significantly on the CPU frequency. How much a decrease in frequency may increase the execution time depends on how much a computation phase is memory bound and on the memory subsystem behavior. For instance, the execution time at the new frequency is not inversely proportional to the change in CPU frequency due to off-chip memory accesses . We used the model proposed in [10] to estimate computation times at reduced speed. The model is based on the following equation:

$$T(f)/T(f_{max}) = \beta(f_{max}/f - 1) + 1 \quad (3)$$

where $T(f)$ is the execution time of running at CPU frequency f, $T(f_{max})$ is the execution time of running at the top frequency $fmax$. The parameter β reflects how much a phase is memory bound. $\beta = 1$ means that halving the frequency doubles the execution time whereas $\beta = 0$ means that a change in frequency does not affect the execution time. We assume $\beta = 0.5$ on average. It is based on measurement results from [20].

3.3 Gear Sets

Two continuous gear sets are used in this work. The first one, unlimited, contains frequencies ranging from 0 to 2.3 GHz. The frequency of 2.3 GHz is assumed to be the highest CPU frequency, per manufacturer's specification. The second continuous set is limited, containing frequencies from 0.8 GHz to 2.3 GHz. The second set is introduced in order to allow comparison with the discrete frequency sets defined below.

We also used several different size gear sets that have from 2 to 15 gears. Furthermore, we tested gear sets with uniform and exponential distribution of frequencies.

The corresponding voltage of 2.3 GHz frequency gear is 1.5 V and for the 0.8 GHz frequency gear the corresponding voltage is 1V. A linear DVFS scenario is assumed as in [3]. Corresponding voltages of other frequency points are values of a linear function determined by points (1V, 0.8 GHz) and (1.5 V, 2.3 GHz). Accordingly, six gears of an evenly distributed set are shown in Table 1 and of the exponentially distributed set in Table 2. The voltage values in continuous case are determined in the same way.

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 1. 6 gear evenly distributed set

Frequency (GHz)	0.8	1.57	1.96	2.15	2.25	2.3
Voltage (V)	1	1.26	1.39	1.45	1.48	1.5

Table 2. 6 gear exponential set

4. Simulation Infrastructure

The observed benchmarks were executed on a PowerPC based cluster with Myrinet interconnection and the execution traces were generated. Paraver [13], a tool to visualize and analyze the execution of parallel applications through traces, was used. We have used the Paraver trace handling environment to extract a trace that corresponds exactly to a period of the iterative application behavior, discarding initialization. It was also used to visualize behavior of observed applications and the degree of load imbalance. Next, Paraver traces were translated to Dimemas trace files. Dimemas [12] is a simulation tool for analysis of the behavior of message-passing applications on a configurable parallel platform.

We have developed a power analysis module to support our study. It works in conjunction with Dimemas to calculates the computation time per MPI process. Our module also assigns a CPU frequency for each process according to the algorithm and the gear set used. Using the assigned frequency, the execution time of a process' computation phase is modified in the Dimemas tracefile. The execution time of a phase is computed according to the time model as the function of the frequency and the β parameter. Dimemas is then used to simulate the modified trace in order to get the execution time of a whole application. The power analysis module computes the original and the new energy/power consumed.

5. Experimental Evaluation

5.1 Metrics

We used two metrics to characterize parallel applications: load balance and parallel efficiency. Load balance of an application is computed as follows:

$$LB = \frac{\sum_{k=1}^{N_{proc}} ComputationTime_k}{N_{proc} * maxComputationTime} \quad (4)$$

where N_{proc} is the number of CPUs, $ComputationTime_k$ is the computation time of k-th processor and $maxComputationTime$ is the maximal value of computation time per process.

If $TotalExecutionTime$ is the execution time of an application, its parallel efficiency is:

$$PE = \frac{\sum_{k=1}^{N_{proc}} ComputationTime_k}{N_{proc} * TotalExecutionTime} \quad (5)$$

.

The results in the paper show the energy needed for executing an application applying DVFS normalized to the original energy when all CPUs run at the top speed. Another widely used metric, the energy-delay product or EDP, is used to account for changes in both the execution time and energy. The EDP is also reported as a normalized value.

5.2 Applications

Several types of benchmarks were used in this work. CG, MG, IS are benchmarks from NAS PB 2.3 [16] and BT-MZ from NAS PB MZ 3.0 suite. Class C benchmarks were used. Also, three real world applications, SPECFEM3D, WRF and PEPC were used. SPECFEM3D [15] simulates seismic wave propagation in sedimentary basin, WRF is numerical weather prediction system [11], and PEPC is a plasma physics code [14]. The applications are executed varying the number of processor from 32 to 128. The number of application processes is stated in a benchmark name, e.g. CG-32.

Potential energy savings in an application depend on its load balance and parallel efficiency. They are shown in Table 3. Note that the values are for an iterative region of an application.

5.3 Results

Recall that the MAX algorithm uses only frequencies less than or equal to the manufacturer specified maximum CPU frequency, while the AVG algorithm allows for possible CPU over-clocking in some of the processors. The MAX algorithm was used in all parametric studies presented in this section. The AVG algorithm was used in 5.3.6 to evaluate whether the overclocking of some processors while running other processors at reduced frequencies can improve both the energy and the execution time.

5.3.1 Different Size Gear Sets

The size and "continuity" of a gear set can have a large effect on the power savings. The CPU energy and EDP values for different gear sets when DVFS is used according to the MAX algorithm are shown in Figure 2. The first column for each frequency set presents the energy normalized to the original value and the second column is a normalized EDP. The first two sets are an unlimited and a limited continuous frequency sets. The rest of the sets are discrete, evenly distributed frequency sets with 2 to 15 gears. The results are given for five applications due to space limitation.

Application	Load balance	Parallel efficiency
BT-MZ-32	35.21%	35.07%
CG-32	97.82%	78.55%
MG-32	94.55%	87.28 %
IS-32	43.77%	8.21%
SPECFEM3D-32	92.80%	92.61%
WRF-32	90.60%	89.53%
CG-64	93.46%	63.36%
MG-64	91.50%	85.60%
IS-64	49.59%	17.00%
SPECFEM3D-96	79.07%	78.65%
PEPC-128	76.12%	67.78%
WRF-128	93.65%	85.27%

Table 3. Application characteristics

These results show that with respect to the energy consumption the unlimited continuous frequency set is better than a limited one only for BT-MZ, IS-32 and IS-64. These three applications need frequencies lower than 0.8 GHz since they have the highest degree of load imbalance. Concerning the size of a gear set, discrete gear sets with six or seven gears are, on average, close to the continuous case, as can be seen in Figure 3. It also shows the effect of load imbalance on energy savings for the unlimited continuous set, two and six discrete gear sets. Even with just two gears it is possible to achieve savings for very load imbalanced applications. SPECFEM3D-32 and both WRFs benchmarks need at least four gears to be able to exploit DVFS in this way, while MG-32 needs even more - six gears. Since CG-32 has the highest degree of load balance, it cannot achieve any energy savings in this way. Other applications save energy with only two or three gears.

One can concluded from the EDP values presented that the execution time of the applications usually is not increased by more than two percent. One exception is PEPC, where execution time in the worst case is increased by 20%. Such an increase in time for PEPC is due to two major computation phases with different load imbalance in one iteration, while only a single DVFS setting is used.

5.3.2 Exponential Gear Sets

In order to explore whether it is possible to achieve similar results with smaller gear set, we have also used five different

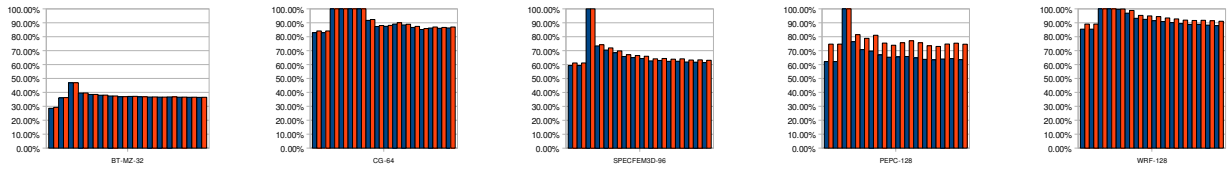


Figure 2. Normalized energy and EDP for different gear sets

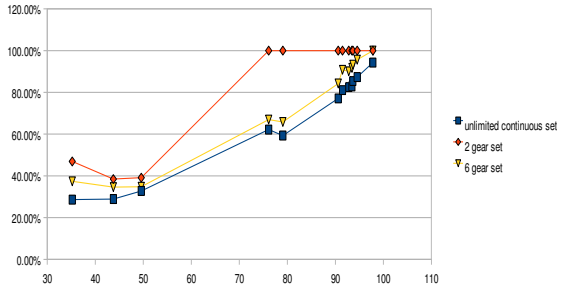


Figure 3. Energy as a function of load balance

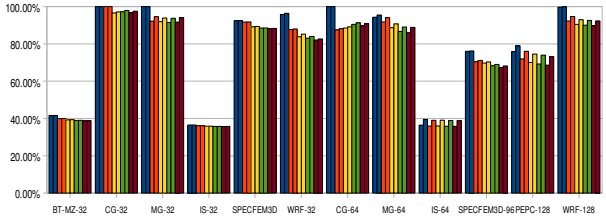


Figure 4. Results for exponential sets

size exponential gear sets. We call a gear set exponential if the difference between two adjacent frequencies is a factor of 2. This distribution of frequencies is expected to show better results for less imbalanced applications since it has greater number of higher frequency gears. The sets used have from three to seven gears.

Figure 4 shows the energy and EDP for each of five sets. When uniformly distributed sets were used, SPECFEM3D-32 and WRFs needed at least four gears to save energy. With exponential sets, they can consume less energy with a three gear set. Similarly, MG-32 needed six uniformly distributed gears to reduce the energy, while with exponential sets four gears are sufficient. Comparing larger gear sets, with six or seven gears, exponentially and evenly distributed gear sets show similar results. Concerning the execution time, it is increased less with exponential sets. Even in the cases of PEPC-128, the execution time is not increased by more than 6.5%.

5.3.3 Effect of Beta

The β parameter indicates how memory bound is a computation phase. Fixed β values from 0.3 to 1 were used. The results in Figure 5 are for the evenly distributed six gear set. The more an application is memory bounded, the higher savings are possible - similar to the sequential case. One can observe that there are applications whose energy varies more for different β values and applications in which the energy is not so much affected by change in β. How much the different values of the β impact the savings of an application depends on its load balance degree. The figure shows that IS-64, SPECFEM3D-96 and PEPC-128 are more sensitive to change in β than others. These are not well balanced applications. Well balanced applications can not always exploit higher memory boundedness due to finite discrete frequency sets and a smaller difference between the original computation and the target time. BT-MZ and IS-32 are the least balanced applications but their energy does not differ a lot for different β values. They need frequencies lower than the lowest one in the set and that is why they can not exploit higher memory pressure.

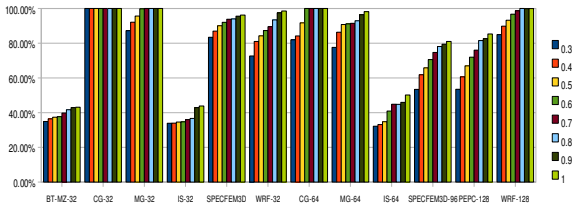


Figure 5. Impact of β parameter

5.3.4 Impact of Static Power

The results presented above used 20% as the fraction of static power at maximum voltage/frequency for computation phases. The impact of static fraction of the total power

is investigated in this section. Figure 6 shows the new total energy varying the static fraction. These results are obtained for uniformly distributed six gear set and the static power percentage varied from 0% to 90% in increments of 10%. When static power makes 70% or more of the total power the energy savings are only a half of those when static power presents 20% of the total power. One can observe from the figure that different applications have different slopes of the energy as a function of the percentage of static in the total power. The slope depends on the load balance of an application - the more load imbalanced it is the higher is the slope.

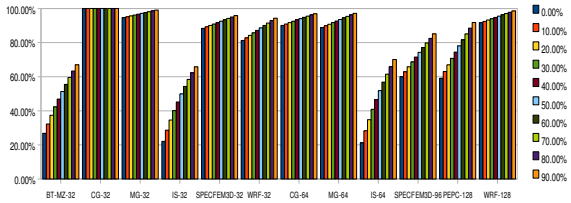


Figure 6. Energy as a function of static power

5.3.5 Activity Factor

The average ratio of computation to communication activity factors, can varies significantly between benchmarks or, for the same benchmark on different platforms. To investigate the impact of the activity factor, it was varied from 1.5 to 3 (based on reports in prior work). The results for the evenly distributed six gear set are presented in Figure 7. The change in energy for different activity factors is dependent on the load balance degree.

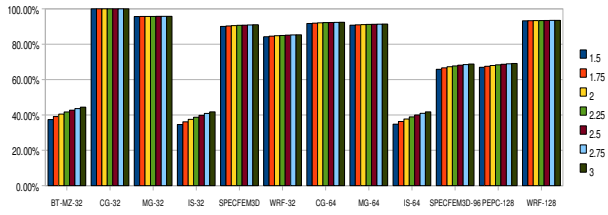


Figure 7. Impact of activity factor

5.3.6 AVG algorithm

Recall that the AVG algorithm can apply a frequency increase in some processors while decreasing frequency in most processors. We have investigated the AVG algorithm allowing the top frequency to be increased by 10% and 20% for the limited continuous frequency set and adding an additional frequency-voltage pair (2.6 GHz, 1.6 V) for discrete evenly distributed six gear set. The results for the continuous case are shown in Figure 8. The energy is reduced for all applications by an amount that depends on the load balance degree. This reduction in energy is between 0.5% for CG-32 and 63% for BT-MZ.

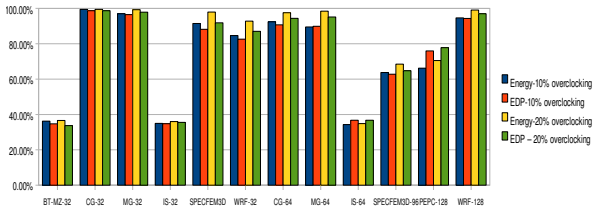


Figure 8. AVG algorithm with continuous set

Normalized energy, EDP and the percentage of processors that need to be overclocked for the discrete case are presented in Figure 9. The AVG algorithm reduces the EDP for all observed applications, except for the most load balanced CG-32 and MG-32. Almost all execution times have decreased. PEPC execution time has increased but less than in the case of the MAX algorithm. The applications with high degree of load imbalance need very few CPUs overclocked (BT-MZ, IS-32, IS-64, PEPC). Some benchmarks, e.g. SPECFEM3D-32, need to overclock 53.13% of its CPUs.

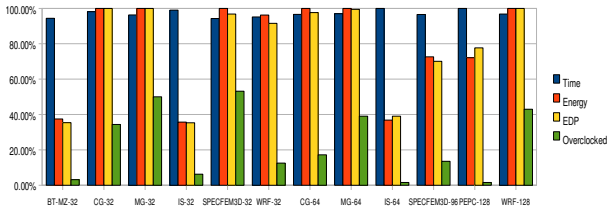


Figure 9. AVG algorithm with discrete set

Comparison of the MAX and the AVG algorithm is depicted in Figure 10. With respect to the CPU energy, the MAX algorithm is better but regarding the execution time the AVG shows better results. Decrease in the execution time reduces energy not only in the CPUs but also in the rest of the system and is important.

6. Conclusions

This paper presented a study of power management for large scale MPI-based load-imbalanced parallel applications. It presented a new algorithm for doing this which applied both DVFS and over-clocking to the CPUs to save

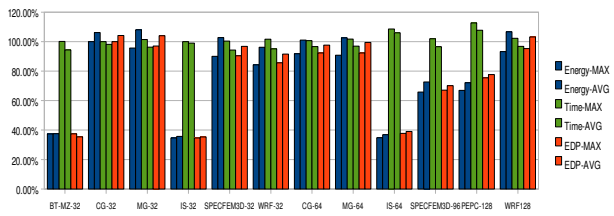


Figure 10. Comparison of MAX and AVG algorithms

energy and to improve execution time. Furthermore, the results show that it is possible to save greater amount of CPU energy than in the case of small clusters due to increased load imbalance of applications. Some applications, for instance IS-32, IS-64 and BT-MZ, are very imbalanced and have a high potential for CPU energy saving, up to 60% .

The proposed new algorithm reduces the execution time in almost all of the applications studied. It achieves CPU energy saving similar to the previously proposed algorithm. Overall, the new algorithm has a higher potential to save overall system energy because it reduces the execution time.

Regarding the gear set size, we can conclude that the gear set with just 6 gears gives good results compared to continuous sets. For well balanced applications and small set size, exponential distribution of frequencies in a set gives better results both the energy and the execution time.

Many other aspects of parallel program execution have an impact on the achievable energy savings. This paper presented an evaluation of the impact of computation to communication activity factor ratio, the memory boundedness in applications, and the fraction of static power on energy savings.

References

[1] C. Boneti, R. Gioiosa, F. J. Cazorla, and M. Valero. A dynamic scheduler for balancing hpc applications. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[2] 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.

[3] 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.

[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] 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.

[6] 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.

[7] 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.

[8] R. Ge, X. Feng, and K. W. Cameron. Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. *sc*, 0:34, 2005.

[9] 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.

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

[11] http://wrf model.org/index.php. Wrf weather prediction system.

[12] http://www.cepba.upc.edu/dimemas/. Dimemas tool web page.

[13] http://www.cepba.upc.edu/paraver/. Paraver tool web page.

[14] http://www.deisa.eu/science/benchmarking/codes/pepc. Pepc application.

[15] http://www.gps.caltech.edu/jtromp/research/downloads.html. Specfem3d sofware package.

[16] http://www.nas.nasa.gov/Resources/Software/npb.html. Nas parallel benchmarks.

[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] 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.

[20] F. Pan, V. Freeh, and D. Smith. Exploring the energy-time tradeoff in high-performance computing. *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 9 pp.–, April 2005.

[21] 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.

[22] J. Rubio, K. Rajamani, F. Rawson, H. Hanson, S. Ghiasi, and T. Keller. Dynamic processor overclocking for improving performance of power-constrained systems. *IBM Research Report*.