# Potential of Energy Saving in MPI Applications with Load Imbalance Using Different Gear Sets

Maja Etinski, Ivan Rodero, Julita Corbalan, Jesus Labarta
Barcelona Supercomputing Center and Technical University of Catalonia
Jordi Girona 29, Nexus 2
08034 Barcelona, Spain
{maja.etinski, ivan.rodero, julita.corbalan, jesus.labarta}@bsc.es

**Abstract.** Dynamic Voltage Frequency Scaling technique is used to scale down CPU performance in order to save energy. We tried to investigate how load imbalance in MPI programs could be exploited to save energy with no or a little penalty in time. In our theoretical study we used a pre-processor of a simulator to changes traces in order to simulate different frequencies. New execution time is simulated by Dimemas using changed traces. Our pre-processor also computes CPU energy consumed relative to the original value. Three different sets of gears are used to compare possible energy savings. Our results show that it is possible to save significant amount of CPU energy exploiting load imbalance in MPI applications. Furthermore, we show that this potential of energy saving does not depend significantly on the size of a gear set.

**Keywords:** power consumption, DVFS, MPI applications

## 1 Introduction

Recently, power consumption became a very important problem in High Performance Computing (HPC) community. Given that Petaflop systems may require 100 megawatts of power, the operating costs are an important reason. That means that at $100 per megawatt, peak operation of this Petaflop machine would be $10,000 per hour [1]. Moreover, there is the problem of reliability because Arrhenius law states that component life expectancy decrease 50% for every $10^0$ C increase in temperature. Since components fail at an annual rate of 2%-3% and a Petaflop system has about 12,000 nodes, a hardware failure happens once every twenty-four hours [1].

The issue of power consumption exists in server/desktop and mobile systems but the main goals are different. HPC systems are used to speedup executions of applications. Their primary goal is performance, not a battery life. Therefore, when investigating possibilities of energy saving in HPC systems, one has to have in mind a constraint of time. It is desirable to have a little or no penalty in time. We also have to consider that a CPU is not always on the critical path and it should be exploited.

CPU power is proportional to the product of the frequency and the square of the voltage. It is very reasonable to try to decrease CPU energy because it presents the main part of overall system energy. The technology that enables the CPU frequency and voltage scaling is called D*ynamic Voltage* and F*requency Scaling* (DVFS). In the bibliography, a pair of a supported frequency and the corresponding voltage is called a processor state (*p-state)* or a gear. For example, the Athlon-64 CPU supports 7 p-states (from 2000 to 800 MHz and the voltage ranges from 1.5V-1.0V). The number of gears can be different for different architectures. Our approach investigates how different gear sets affect the possibilities of energy saving.

There is an existent work performed in the field of energy-time tradeoff in MPI programs but it assumes direct measurements by multimeters, as it is done for example in [2]. They determine the instantaneous power and integrate it over time to determine the consumed energy. For our study we do not need exact value of consumed energy because our aim is to explore the benefits of the DVFS technique using different frequency sets. Thus, relative values are enough for the comparisons.

This paper presents a theoretical study to evaluate the potential benefit of exploiting load imbalance in MPI programs. This is the one of potentials for energy saving under assumption that the execution time should not be increased more than few percents. Others are memory bound tasks and communication intensive programs but they are not considered in this paper.

Our study is based on simulations and consists of the next steps. First, we obtain traces of each application using different numbers of MPI processes. We used NAS Parallel Benchmark Multi Zone as a benchmark suite. After determining the application load imbalance we pre-process the traces in order to simulate different frequencies. The pre-processor uses three algorithms to determine a new frequency of each process (for the whole execution). The algorithms will be described later. Then, we simulate the same execution but using the new CPU frequencies. Finally, the post-processor calculates metrics of the original execution and the new simulated one.

Our work has two main contributions. The first one is to show the potential of energy saving in load imbalanced MPI applications. The results show that is possible for very load imbalanced application to save up to 70% of CPU energy using DVFS when the application runs on 16 CPUs. The other contribution is that we found that gear sets with much more gears do not provide more benefits than sets with less gears.

The rest of this paper is organized as follows. In section 2 we present the related work. Next, in section 3, we describe in details the work done. It is followed by results of our simulations in section 4. Finally, in section 5, we expose our conclusions and plans for future work.

## 2 Related Work

As we mentioned above, the majority of the work done in the field of energy saving with MPI programs uses direct measurements as an evaluation technique of energy consumed.

In [3], Kappiah *et al.* developed a system called Jitter that should exploit *inter-node bottleneck* in MPI programs. In other words, some of the nodes arrive earlier at a synchronization point and these nodes could execute in lower p-states without penalty in time. They assume iterative applications and that the iterations are relatively stable. Past behaviour is used to predict future behaviour. Jitter saves 8% energy (system energy) and increase execution time by 2.6% while running Aztec from the Purple suite on 8 nodes.

In [4], Lim *et al.* present an MPI runtime system that dynamically reduces the CPU performance during communication regions in MPI programs without user involvement. The assumption is that during communication regions the CPU is not on the critical path.

In [5], Freeh et al. explore energy consumption and execution time across different numbers of nodes. All measurements are done using single gear for whole execution. They vary the number of nodes and p-state for an application. Then, they compare execution time and consumed energy. They found that, for some programs, is possible to both consume less energy and execute in less time when using a larger number of nodes, each at reduced state. In the second part of the paper, a model is developed in order to predict execution time and energy consumed for an application running on 32 node cluster. The cluster is not power scalable. They predict what would be execution time and energy consumed of an application running at lower gears. Execution time and energy consumed at top state are used, as well as those values for smaller numbers of nodes at different gears.

Rountree *et al.* developed a system in [6] 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 some constraints.

In [2], the idea is to divide programs into phases and then assign different gears to phases according to previous measurements and a heuristic. The authors found that there are three groups of applications Single gear benefit presents group of applications that benefit from using a single lower gear but they do not show significant benefit from multiple-gear solution. Applications that benefit from multiple-gear solution are multiple gear solution. There are also no benefit applications.

# 3 Methodology

## 3.1 Visualization and Simulation Tools

Paraver [7] is a tool used to visualize and analyse the execution of parallel applications through traces. We determined the degree of load imbalance in our benchmarks using Paraver so afterwards we can analyze the impact of load imbalance on the energy saving potential. Dimemas is a simulation tool for the parametric analysis of the behaviour of message-passing applications on a configurable parallel platform [8].

## 3.2 Traces and Simulation Process

The flow of our study is graphically presented in figure 1. First, we execute the benchmarks and traces are generated with BSC instrumentation tools. The applications are executed on MareNostrum supercomputer that comprises 2560 JS21 compute nodes and 42 p615 servers. Every node has four IBM Power PC 970MP processors at 2.3 GHz processors running Linux operating system with 8 GB of memory RAM. A Myrinet network is used for parallel applications communications. Then, we used the pre-processor that translates the Paraver trace file to a Dimemas trace file. The Dimemas trace file contains lines describing tasks that are CPU bursts between MPI calls. These lines have information about the task duration and the corresponding MPI process identification number. Moreover, the pre-processor calculates the total sum of task durations per MPI process to determine maximal, average and minimal sum of task durations per process. New frequency is assigned to each process according to one of the three algorithms evaluated. We assume that it is possible to independently change the frequency per CPU. The new CPU frequency of a process will be used during the simulation as a frequency of the process for whole execution.

We have evaluated three algorithms. The first algorithm is AVG, it assigns new frequency to a process so that the new sum of task durations of the process is equal to the average value of sums of task durations per process. We do not take into account how much is a program memory bound. If the original task duration is $t_1$ then the new task duration is $t_2=( f_1 / f_2 )* t_1$ where $f_1$ and $f_2$ are original and new frequencies respectively. MAX scales the frequency of each task so that the new sum of task durations per each process is equal to the maximal value of original sums. The last algorithm MIN scales frequencies of processes in order to achieve that the sum of task durations per process is equal to the minimal value of original sums. This algorithm assumes that is possible to use higher frequencies than the frequency used during the execution of a program. The first two algorithms are realistic but the third one is used only to get a feeling of how much additional energy is necessary to speedup a program.

Although, it is not possible to scale frequency to an arbitrary value frequencies are not scaled exactly according to explained algorithms. A new frequency is the closest of the existing frequencies to the frequency that should be assigned according to the algorithm. Sets of gears that we used are presented in tables 1-3. The first set of gears includes three gears with frequencies higher than the original one, 2300 MHz. The second one is inspired by AMD Athlon 64 processor. The last one has only three gears. We used MIN algorithm only for the first set of gears because it uses frequencies higher than the top one. The first set has more gears than existing processors have. It should imitate continuous frequency set in order to obtain a feeling of how much energy would be possible to save without limitation of small gear set.

Figure1. Scheme of the methodology

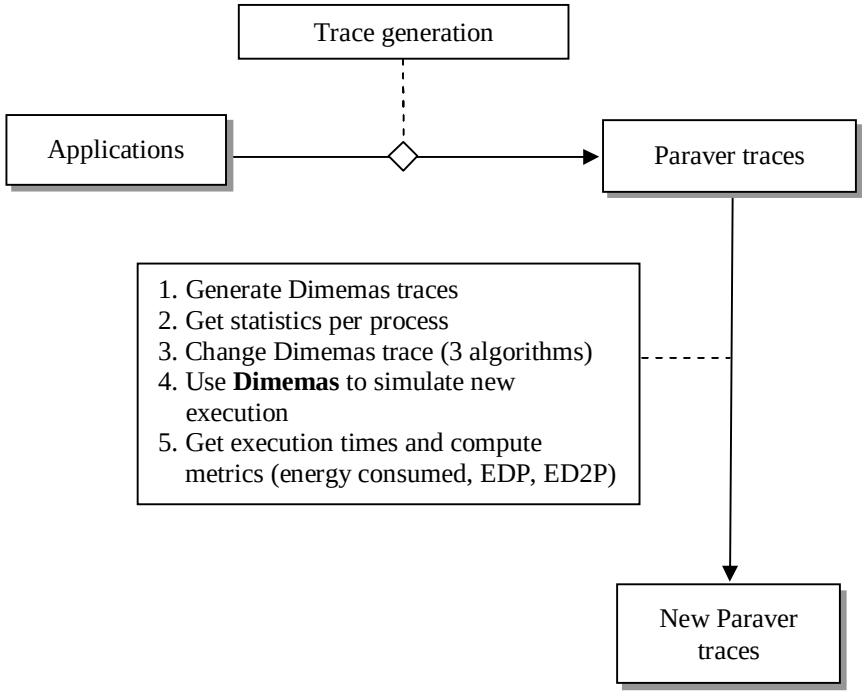


Table 1. The first set of gears

| Frequency(MHz) | Voltage(V) |
|---|---|
| 2600 | 1.65 |
| 2500 | 1.60 |
| 2400 | 1.55 |
| **2300** | **1.50** |
| 2200 | 1.47 |
| 2100 | 1.43 |
| 2000 | 1.40 |
| 1900 | 1.37 |
| 1800 | 1.33 |
| 1700 | 1.30 |
| 1600 | 1.27 |
| 1500 | 1.23 |
| 1400 | 1.20 |
| 1300 | 1.17 |
| 1200 | 1.13 |
| 1100 | 1.10 |
| 1000 | 1.07 |
| 900 | 1.03 |
| 800 | 1.00 |

Table 2. The second frequency set

| Frequency(MHz) | Voltage(V) |
|---|---|
| **2300** | **1.5** |
| 2100 | 1.4 |
| 1900 | 1.35 |
| 1600 | 1.3 |
| 1300 | 1.2 |
| 1000 | 1.1 |
| 800 | 1.0 |

Table 3. The third frequency set

| Frequency(MHz) | Voltage(V) |
|---|---|
| **2300** | **1.5** |
| 1300 | 1.2 |
| 800 | 1.0 |

In the next step the pre-processor generates a new Dimemas trace file. The pre-processor simulates new frequencies by changing task durations inversely proportional to the change of frequency. Lines that do not present CPU bursts are not changed. Dimemas takes as inputs generated trace file and a configuration file that describes MareNostrum nodes used to execute the program. Dimemas simulates the program execution and generates new Paraver trace file. Dimemas simulate communication using the original frequency but it does not affect the communication time significantly, as it is shown in [4]. As output of the simulation we obtain a Paraver trace file that helps us to extract the new execution time.

The post-processor computes the energy consumed. It separately computes energy consumed during CPU bursts and in MPI calls. The total CPU power is sum of dynamic and static power. With regard to the fact that dynamic power is many times greater than static, we model CPU power as dynamic power. Although we do not know activity factors of our benchmarks, we can not compute absolute values of consumed energy. But assuming that the activity factor is approximately the same for whole execution of one application we can use relative values in our comparisons. In [9] authors also assumed gear set and estimated relative power saving per multi-core

processor using the fact that power is proportional to the product of frequency and square of voltage. To validate this estimation they incorporated DVFS into IBM´s Turandot simulator. The estimations were almost equal to the results of simulation for all tested benchmarks.

Our results present relative values of energy consumed normalized to the energy consumed during original execution. They also include the original execution times and the new execution times that are result of the simulation. Although the difference between real execution time and simulated one are very small (usually less than 1%), in order to perform a more realistic comparison we also used simulated execution time instead of real original execution time. Original traces are simulated with Dimemas and that execution time is used.

EDP (Energy Delay Product) and ED2P (Energy Delay Square Product) metrics are used as scalar metrics of two dimensional tradeoff. The first metric is product of energy consumed and execution time $EDP = E * t$. In HPC community, performance is primary goal so the second metric is used to emphasize even more execution time $ED2P = E * t * t$.

## 4 Simulation Results

### 4.1 Applications

NAS PB MZ 3.0 is used as a benchmark suite [10]. The number of OpenMP threads is always set to one, so all applications are only MPI.The application benchmarks Lower-Upper Symmetric Gauss-Seidel (LU), Scalar Penta-diagonal (SP), and Block Tri-diagonal (BT) solve discretized versions of the unsteady, compressible Navier-Stokes equations in three spatial dimensions. Although the class A is used, they are run on 2, 4, 8 and 16 CPUs.

We generated Paraver trace files and we used them to determine the degree of load imbalance in our applications. BT is very load imbalanced program. On the contrary, benchmarks LU and SP are load balanced programs. For example, in 8 MPI process BT a process that runs the least runs 7.25% of execution time. The most running process runs 97.99% of execution time. The least running process of 8 MPI processes LU runs 79.14% and the most running runs 96.00% of execution time. SP is the most load balanced application of these three. Its processes running times varies from 91.33% to 95.41%.

### 4.2 Results

As we mentioned above, in our research we used 2, 4, 8, and 16 MPI processes. The algorithm MIN is applied only for the first set of gears because the first set includes frequencies higher than the original frequency. If we would apply this algorithm in cases of the others frequency sets there would be no difference comparing to original executions. All the results shown in tables 4, 5 and 6 present values relative to values of the original execution with corresponding number of MPI processes.

Table 4. Results of the first gear set

| Num of MPIs | Metric | AVG BT | MAX BT | MIN BT | AVG LU | MAX LU | MIN LU | AVG SP | MAX SP | MIN SP |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Time | 0.88 | 1.00 | 0.88 | 1.00 | 1.02 | 0.98 | 1.00 | 1.00 | 1.00 |
| | Energy | 072 | 0.58 | 1.05 | 1.00 | 0.98 | 1.03 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.64 | 0.58 | 0.93 | 1.00 | 1.00 | 1.01 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.56 | 0.58 | 0.82 | 1.00 | 1.02 | 0.99 | 1.00 | 1.00 | 1.00 |
| 4 | Time | 0.89 | 1.00 | 0.88 | 1.00 | 1.01 | 0.97 | 1.00 | 1.02 | 0.97 |
| | Energy | 0.73 | 0.43 | 1.13 | 1.00 | 0.97 | 1.02 | 1.00 | 0.98 | 1.03 |
| | EDP | 0.65 | 0.43 | 1.00 | 1.00 | 0.98 | 0.99 | 1.00 | 1.00 | 1.01 |
| | ED2P | 0.58 | 0.43 | 0.88 | 1.00 | 0.99 | 0.96 | 1.00 | 1.02 | 0.98 |
| 8 | Time | 0.91 | 1.03 | 0.89 | 0.97 | 1.03 | 0.88 | 1.00 | 1.02 | 0.98 |
| | Energy | 0.63 | 0.30 | 1.17 | 0.98 | 0.93 | 1.14 | 1.02 | 0.98 | 1.03 |
| | EDP | 0.57 | 0.31 | 1.04 | 0.95 | 0.95 | 1.01 | 1.02 | 1.00 | 1.01 |
| | ED2P | 0.52 | 0.31 | 0.92 | 0.92 | 0.98 | 0.90 | 1.02 | 1.01 | 0.98 |
| 16 | Time | 0.97 | 1.11 | 0.88 | 0.99 | 1.03 | 0.90 | 1.00 | 1.05 | 0.96 |
| | Energy | 0.69 | 0.26 | 1.19 | 0.99 | 0.95 | 1.13 | 1.00 | 0.96 | 1.12 |
| | EDP | 0.66 | 0.29 | 1.05 | 0.98 | 0.98 | 1.02 | 1.01 | 1.01 | 1.07 |
| | ED2P | 0.64 | 0.32 | 0.93 | 0.97 | 1.02 | 0.92 | 1.01 | 1.06 | 1.02 |

Table 5. Results of the second gear set

| Num of MPIs | Metric | AVG BT | MAX BT | AVG LU | MAX LU | AVG SP | MAX SP |
|---|---|---|---|---|---|---|---|
| 2 | Time | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Energy | 0.62 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.62 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.62 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| 4 | Time | 1.01 | 1.07 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Energy | 0.63 | 0.45 | 1.00 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.64 | 0.48 | 1.00 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.64 | 0.52 | 1.00 | 1.00 | 1.00 | 1.00 |
| 8 | Time | 1.03 | 1.08 | 1.00 | 1.04 | 1.00 | 1.04 |
| | Energy | 0.62 | 0.31 | 0.95 | 0.92 | 1.00 | 1.02 |
| | EDP | 0.64 | 0.33 | 0.95 | 0.95 | 1.00 | 1.06 |
| | ED2P | 0.65 | 0.36 | 0.95 | 0.99 | 1.00 | 1.11 |
| 16 | Time | 1.07 | 1.17 | 1.02 | 1.07 | 1.03 | 1.07 |
| | Energy | 0.62 | 0.27 | 0.99 | 0.94 | 1.02 | 0.97 |
| | EDP | 0.67 | 0.32 | 1.01 | 1.00 | 1.05 | 1.04 |
| | ED2P | 0.73 | 0.37 | 1.04 | 1.07 | 1.09 | 1.12 |

Table 6. Results of the third gear set

| Num of MPIs | Metric | AVG BT | MAX BT | AVG LU | MAX LU | AVG SP | MAX SP |
|---|---|---|---|---|---|---|---|
| 2 | Time | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Energy | 0.68 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.68 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.68 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| 4 | Time | 1.01 | 1.07 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Energy | 0.63 | 0.45 | 1.00 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.64 | 0.48 | 1.00 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.64 | 0.52 | 1.00 | 1.00 | 1.00 | 1.00 |
| 8 | Time | 1.03 | 1.08 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Energy | 0.65 | 0.31 | 1.00 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.65 | 0.33 | 1.00 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.68 | 0.36 | 1.00 | 1.00 | 1.00 | 1.00 |
| 16 | Time | 1.08 | 1.17 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Energy | 0.57 | 0.27 | 1.00 | 1.00 | 1.00 | 1.00 |
| | EDP | 0.61 | 0.32 | 1.00 | 1.00 | 1.00 | 1.00 |
| | ED2P | 0.66 | 0.37 | 1.00 | 1.00 | 1.00 | 1.00 |

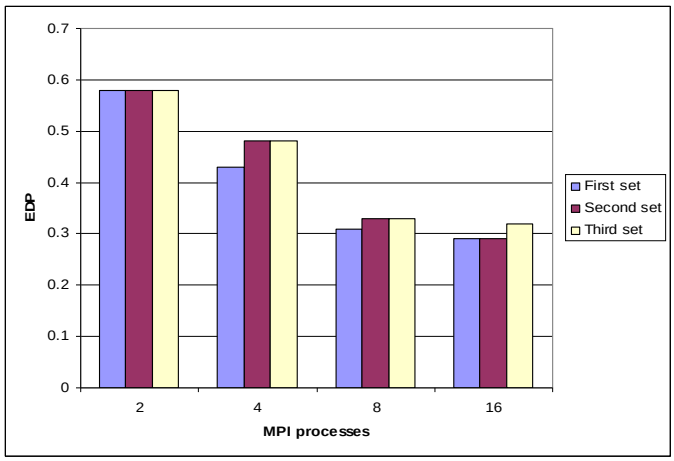
### 4.3 Discussion of Results

MIN algorithm is applied only for the first gear set. It diminishes execution time but it increase energy consumed. Applying AVG and MAX algorithms to BT benchmark gives always EDP less than 1. MAX algorithm saves more energy than AVG but it has greater penalty in time. The penalty reaches 17% in the case of the second and third frequency sets for 16 MPI processes. This penalty is due to the fact that our algorithms assign frequency to CPU for whole execution. One of processes of BT has longer initialization part than others although it runs much less than some others processes. When its performance is reduced others have to wait for it to finish initialization part. Although class A is used this relatively small change in seconds affects relative value of execution time. This penalty in time could be avoided by changing frequencies after initialization part i.e. before iterations. Scientific applications usually have similar structure i.e. they are iterative. When the number of MPI processes increases, it is obtained the higher relative energy saving for the BT benchmark with MAX algorithm

Results for LU and SP benchmarks are more similar to originals than results of BT. This is because of the load imbalance of the BT benchmark. Due to high degree of load balance of LU and SP, maximal energy saving is 8% of CPU energy. Regarding the last gear set, all assigned frequencies for LU and SP are equal to the original one.

Results show that there is no significant advantage in using greater gear sets. The third set with three gears also gives very good results as we can see in figure 2.

Figure 2. Relative EDP values for BT benchmark with MAX algorithm



## 5 Conclusions and Future Work

Our work aims to exploit load imbalance in programs. There are potentials of energy saving in imbalanced MPI programs. Among tested algorithms, MAX algorithm gives best results. Also, there is no need to use large gear sets.

Our future works includes the evaluation of real applications and NAS PB suite with larger numbers of MPI processes. Also, we are considering the integration of OPM (operations per miss) as a measure of memory pressure. It is used in [2] to approximate memory pressure. In that case our simulation of different frequencies would be even more accurate.

## Acknowledgments

This work has been supported by the Spanish Ministry of Science and Technology under contract TIN2007-60625 and by the BSC-IBM MareIncognito research agreement. We give spacial thanks to Kamil Kedzierski for his comments concerning power.

## References

1. Chung-hsing Hsu and Wu-chun Feng. A power-aware run-time system for high-performance computing. *In Supercomputing*, November 2005.
2. V. Freeh, D. Lowenthal, F. Pan, and N. Kappiah. Using multiple energy gears in MPI programs on a power-scalable cluster. *Proc. Symp. Principles and Practices of Parallel Programming (PPoPP 05)*, June 2005.
3. N. Kappiah, V. Freeh, D. Lowenthal. Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in MPI programs. *In Supercomputing*, November 2005.
4. M.Y. Lim, V.W. Freeh, D.K. Lowenthal . Adaptive,Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs, *In Supercomputing*, November 2006.
5. V. Freeh, D. Lowenthal, R. Springer, F. Pan, and N. Kappiah. Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster. *Proc. 19th Int'l Parallel &Distributed Processing Symp. (IPDPS 05)*, Apr. 2005.
6. B. Rountree, D. Lowenthal, S. Funk, V. Freeh, B. Supinski, M. Schulz. Bounding Energy Consumption in Large-Scale MPI Programs. , *In Supercomputing*, November 2007.
7. Paraver tool web page, http://www.cepba.upc.edu/paraver/
8. Dimemas tool web page, http://www.cepba.upc.edu/dimemas/
9. C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose, M. Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget, *39th ACM/IEEE International Symposium on Microarchitecture (MICRO-39)* December, 2006.
10. NAS Parallel Benchmarks, Multi-Zone Versions, http://www.nas.nasa.gov/News/Techreports/2003/PDF/nas-03-010.pdf