Combining Load Balancing and DVFS to Save Energy on Imbalanced Applications *

Edson L. Padoin[1,2], Laércio L. Pilla[1,3], Márcio Castro[1,3], Philippe O. A. Navaux[1], Jean-François Mehaut[4]
{elpadoin,navaux}@inf.ufrgs.br, {laercio.pilla,mbcastro}@ufsc.br, jean-francois.mehaut@imag.fr

[1]Institute of Informatics
Federal University of Rio Grande do Sul (UFRGS) – Porto Alegre, RS – Brazil
[2]Department of Exact Sciences and Engineering
Regional University of Northwest of Rio Grande do Sul (UNIJUI) – Ijui, RS – Brazil
[3]Department of Informatics and Statistics
Federal University of Santa Catarina (UFSC) – Florianópolis, SC – Brazil
[4]Grenoble Informatics Laboratory (LIG)
University of Grenoble – Grenoble – France

1. Introduction and Motivation

Parallel scientific applications allow addressing grand challenges in science. Scientific applications that represent natural phenomena, such as molecular dynamics simulations, or forecast natural phenomena, such as the propagation of seismic waves and the weather, are run in High Performance Computing (HPC) platforms aiming at more accurate results and smaller execution times.

However, these applications have increasing amounts of data to be processed, demanding a larger number of processing cores, and spend enormous amounts of energy to run. To attend to this demand, supercomputers are scaling their performance exponentially over the years leading to an exponential growth in power consumption [1].

Whereas some scientific applications have a load that is balanced, others are considered irregular applications due to the fact that they have tasks with different processing demands, which makes it difficult to use all available resources at the hardware level efficiently.

In this work we focus on an energy-aware approach to save energy when running these irregular applications. We propose a new load balancing (LB) strategy named ENERGYLB to save energy by reducing the average power demand of cores.

* This work was partially supported by CNPq, FAPERGS, FINEP, CAPES (under grants 3471-13-6, 5854-11-3, and 5847-11-7) and by the HPC-GA research project, which has received funding from the European Community's Seventh Framework Programme (FP7-PEOPLE) under grant agreement number 295217. This work was developed in the context of LICIA, an associated international laboratory between UFRGS and University of Grenoble.

2. Energy-saving approach

When a system runs load imbalanced applications, cores with shorter tasks finish first and remain spending energy without doing any actual work for the application, i.e., idle cores are wasting energy. These cores could perform their tasks slower and end at the same time as the other cores. Thus, one possible approach to save energy without incurring on a large growth of the execution time relies on the use of Dynamic Voltage and Frequency Scaling (DVFS).

From this premise, to save energy without losing performance we designed a new load balancing algorithm named ENERGYLB to be used with load imbalanced applications. ENERGYLB is a periodic load balancer that uses DVFS to change cores' frequency during the application execution and not when execution ends like some mechanisms.

First, ENERGYLB gets information from all cores used by the application. It then weights the core loads according to their current clock frequencies as show in Equation 1.

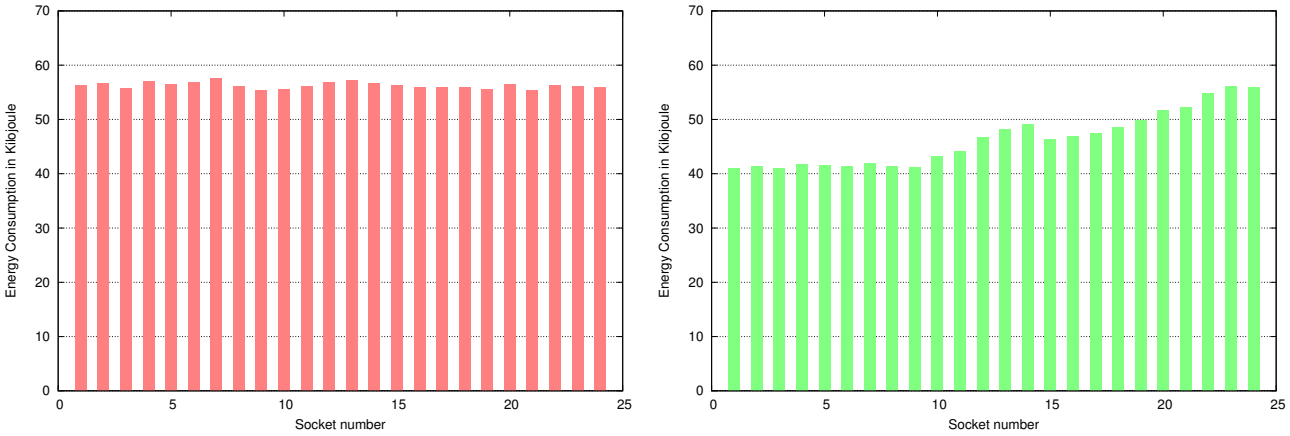
$$Wload_i = current\_load_i / current\_freq_i \quad (1)$$

Once the weights of all cores are computed, the load imbalance is computed as show in Equation 2.

$$li = Wload_{max} / Wload_{min} \quad (2)$$

where $Wload_{max}$ is the load of the most overloaded core, and $Wload_{min}$ is the load of the least loaded core.

When load imbalance is detected, the difference between the most and least loaded cores is compared to the available clock frequency limits. If load imbalance is greater than the



(a) noLB

(b) with ENERGYLB

Figure 1. Energy Consumption (in Kjoule)

ratio between the highest and lowest frequency, then ENERGYLB benefits from the execution time-focused load balancers available with CHARM++ [5]. In this case, reducing only the frequency would not be enough to balance the load, so calling other load balancing strategies helps by migrating tasks.

Otherwise, ENERGYLB maintains the frequency of the cores with higher load and decreases the frequency of the other cores according to their relative loads through DVFS.

By updating the frequency of the cores according to their relative loads, ENERGYLB reduces the average power demand of the system and, consequently, saves energy.

3. Energy Monitoring Tool

Current platforms have different interfaces to collect power and energy data from different equipment. In some tools, the collected data can be analyzed only after the execution of tests making it difficult to correlate performance with power and energy consumption. In order to provide a single solution to measure instantaneous power and energy consumption during the execution of applications on homogeneous and heterogeneous systems, we developed a new tool named Energy Daemon Tool. It works in two consecutive phases when executed along with applications, as described below.

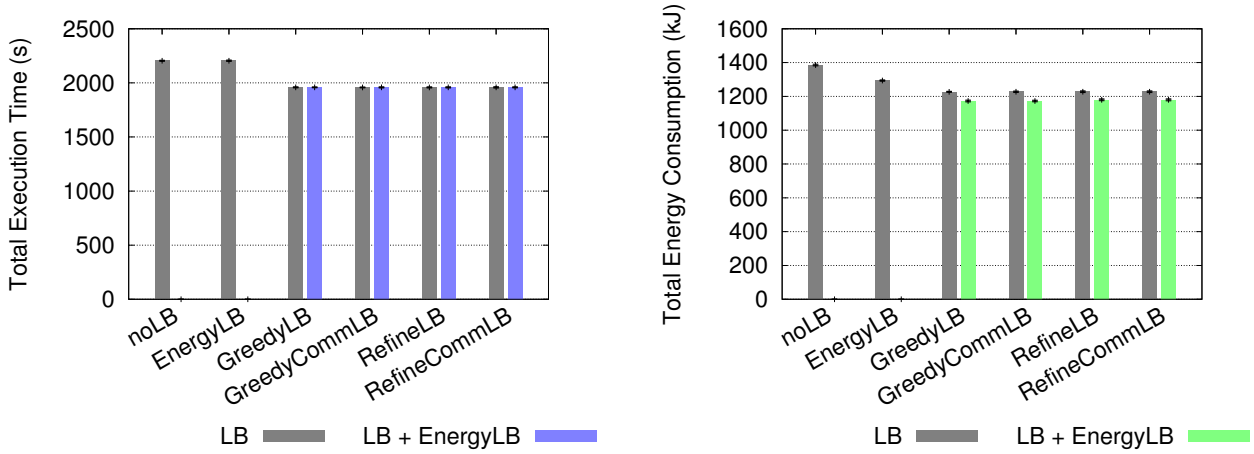
- **Discovery phase:** at the very beginning, the tool obtains mostly static information from the platform, such as processor manufacturer, processor model, available clock frequencies and current clock frequency. This information will guide the decisions on how power and energy data will be collected from the underlying platform.
- **Monitoring phase:** during the execution of the application, Energy Daemon Tool monitors the system to collect power or energy information with a periodicity defined by the user. It maintains some statistical data such as the minimum, maximum and average power, and the energy consumption. In addition to that, the tool can also trace instantaneous power and clock frequency of the processors during execution. In this case, data is stored in output files that can be visualized with standard graphing utilities such as Gnuplot.

In this paper, the MSR registers available on Intel Sandy Bridge processors were used to measure energy consumption.

4. ENERGYLB Evaluation

We conducted our experimental evaluation on an SGI Altix UV 2000 system composed of 24 Intel Xeon E5-4640 Sandy Bridge-EP processors with 8 physical cores. In these processors, there are 14 clock frequency levels available, varying from 1.2 GHz to 2.4 GHz. However, current processors do not allow DVFS on each core individually. So, in this work, tests were performed using 24 cores only (one on each socket). This approach was used because in Intel processors, although it is possible to set the frequency of each core individually, no power gains are achieved due to a shared voltage rail and clock source.

For the evaluation of ENERGYLB, the CHARM++ programming model was used. CHARM++ is a parallel programming system that provides a set of information about tasks and the application [5], benchmarks and load balancers to migrate objects between processors [2, 3]. We selected a benchmark named ComprehensiveBench developed to Charm++. To perform the tests, the load balancers



(a) Execution Time

(b) Energy Consumption

Figure 2. Execution Time and Energy Consumption to different load balancers

GreedyLB, *GreedyCommLB*, *RefineLB* and *RefineCommLB* were selected.

The operating system used in the machine is a UV2000 GNU/Linux distribution with kernel version 3.0.74. The application benchmark was compiled using gcc version 4.3 and the version 6.5.1 of the platform Charm++. Results show the average of a minimum of 10 runs. The relative error is less than 5% using a 95% statistical confidence with Students t-distribution.

Figure 1 illustrates the energy consumption by socket measured for ComprehensiveBench with 200 chares [3], 250 timesteps, and minimum and maximum task loads of 5 ms and 800 ms, respectively. Load balancing frequency was set to 10 iterations.

Figure 1(a) shows the energy consumption for different sockets when no LB algorithm (noLB) was used. In all sockets we have a similar energy consumption. However, when using ENERGYLB, which computes the load imbalance and changes the clock frequency used (see Figure 1(b)), we are able to save energy 236.7 Joules.

Figure 2 illustrates the execution time and energy consumption measured by the Energy Daemon Tool for ComprehensiveBench. First, we present the execution time and energy consumption for a baseline execution where no LB algorithm was used (noLB), followed by the energy measured when using ENERGYLB only. Finally we compare the energy consumption when using CHARM++ LBs and them in conjunction with ENERGYLB.

When using ENERGYLB, the average power demand measured during the execution was reduced from 26 W to 24.4 W in comparison to others LB. This reduction occurs because ENERGYLB sets the frequency of some overloaded processors to the maximum value available, resulting in a higher power demand of these processors. Analogously, the frequency of underloaded processors is set to the minimum value possible (1.2 GHz) or a relative value, resulting in several processors demanding less power, and thus, saving energy. Experimental results with this benchmark present energy savings between 4% and 13% when ENERGYLB was used.

Others test were also performed using two variants of ENERGYLB in [4]. The first one, called *Fine-Grained EnergyLB (*FG-ENERGYLB*)*, is suitable for platforms composed of few tens of cores that allow per-core DVFS. The second one, called *Coarse-Grained EnergyLB (*CG-ENERGYLB*)*, is suitable for current HPC platforms composed of several multi-core processors that feature per-chip DVFS.

5. Conclusion

In this paper, we presented our initial efforts to combine load balancing and DVFS in order to save energy. Our approach to global scheduling benefits from CHARM++'s load balancing framework and enables the reduction of power demand during the execution of imbalanced applications.

Future efforts focus on: implementing more robust mechanisms to decide when to balance load and/or change clock frequencies; developing a hierarchical algorithm to use all available cores in the platform; and validating the algorithms with different applications and platforms.

References

[1] Y. Dong, J. Chen, and T. Tang. Power measurements and analyses of massive object storage system. In *International*

Conference on Computer and Information Technology (CIT), pages 1317–1322. IEEE, 2010.

[2] L. V. Kale, E. Bohm, C. L. Mendes, T. Wilmarth, and G. Zheng. Programming petascale applications with CHARM++ and AMPI. *Petascale Computing: Algorithms and Applications*, 1:421–441, 2007.

[3] L. V. Kale and S. Krishnan. CHARM++: A portable concurrent object oriented system based on C++. In *Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 91–108. ACM, 1993.

[4] E. L. Padoin, M. Castro, L. L. Pilla, P. O. A. Navaux, and J.-F. Mehaut. Saving energy by exploiting residual imbalances on iterative applications. In *21st International Conference on High Performance Computing (HiPC) (awaiting publication)*, pages 1–10, Goa, India, 2014.

[5] G. Zheng, A. Bhatelé, E. Meneses, and L. V. Kalé. Periodic hierarchical load balancing for large supercomputers. *International Journal of High Performance Computing Applications*, 25(4):371–385, 2011.