Using Power Demand and Residual Load Imbalance in the Load Balancing to Save Energy of Parallel Systems

Edson Luiz Padoin, Philippe O.A. Navaux, Jean-François Méhaut

Using Power Demand and Residual Load Imbalance in the Load Balancing to Save Energy of Parallel Systems

Edson Luiz Padoin[1], Philippe O. A. Navaux[2], and Jean-François Méhaut[3]

[1] Regional University of Northwest of Rio Grande do Sul (UNIJUI), Ijuí, RS – Brazil
padoin@unijui.edu.br
[2] Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS – Brazil
navaux@inf.ufrgs.br
[3] University of Grenoble, Grenoble – France
Jean-Francois.Mehaut@imag.fr

Abstract

The power consumption of the High Performance Computing (HPC) systems is an increasing concern as large-scale systems grow in size and, consequently, consume more energy. In response to this challenge, we have develop and evaluate new energy-aware load balancers to reduce the average power demand and save energy of parallel systems when scientific applications with imbalanced load are executed. Our load balancers combine dynamic load balancing with DVFS techniques in order to reduce the clock frequency of underloaded computing cores which experience some residual imbalance even after tasks are remapped. The results show that our load balancers present power reductions of 7.5% in average with the fine-grained variant that performs per-core DVFS, and of 18.75% with the coarse-grained variant that performs per-chip DVFS over real applications.

Keywords: Energy Consumption, Load Balancing, DVFS, Power Demand, Energy Saving

1 Introduction

The main focus of HPC systems has been performance, and nowadays the HPC community works toward building Exascale systems, which will provide unprecedented computational power, allowing to solve even larger scientific problems. However, if we conceive Exascale supercomputers by scaling the current technology, these systems could demand over a Gigawatt [8, 4]. In this way, a global research effort has risen to try to break this barrier while avoiding such high power demands.

Reductions in the total execution time of real applications are also relevant in the perspective of energy consumption, as energy is saved when hardware resources are used for a shorter time. However, it is possible to achieve even greater energy savings if the runtime system is

able to exploit the residual imbalances to fine tune the voltage and frequency of cores accordingly. In this case, a challenge lies in reducing the energy consumption of the application while maintaining a similar performance.

Scientific applications generally have an irregular computational load and tasks with different processing demands, which makes the load distribution among the processors more difficult. In this context, different approaches have been used to mitigate these imbalanced workloads and achieve an efficient use of all available parallel resources. However, most of these load balancing strategies focus only on reducing the execution time using only computational load, neglecting power demand and total energy consumption.

Generally, approaches to reduce the power demand are used separately of load balancing research for increasing application's performance. Several proposal have used DVFS, however this technique may cause performance degradation and an increase of the total execution time of parallel applications. Other proposals have used load balancing strategies to mitigate the imbalanced workloads and achieve an efficient use of all parallel resources, which reduce the overall execution time, and, consequently, save energy. Our developed approaches make the case that both strategies, load balancing and DVFS, could be used together over real applications to save energy. They identify the possibility of reducing the processors clock during the runtime to achieve better gains over other load balancer algorithms. In this form, energy improvements are achieved due the reduction of average power demand during the execution and also through of reducing of the execution time by reducing the amount of tasks migrated.

In this paper we propose and validate over real applications our energy-aware load balancers, which minimize the total energy consumption when applied over iterative applications with imbalanced loads. We show that, combining load balancing with DVFS, our energy-aware load balancers are able to reduce the energy spent with load balancing when they are used over other state-of-the-art algorithms.

The remaining sections of this paper are organized as follows. Section 2 discusses some of the related works on DVFS and energy-aware load balancing. The main concepts of our energy-ware load balancer (EnergyLB) as well as its two variants and implementation details are discussed in Section 3. In Section 4 we present the evaluation methodology and the real applications used in the conducted experiments. In addition, in Section 5 we address the results obtained from the experiments. Finally, the Section 6 emphasizes the scientific contribution of the work and notes several challenges that we can address in the future.

2 Related Work

Different approaches are used to reduce the power demand of the HPC systems aim improve their energy efficiency. In this way **Dynamic voltage and frequency scaling (DVFS)** has been used in different contexts as a means to save energy.

Spiliopoulos *et al.* [15] extended the Gem5 simulator to support full-system DVFS modeling and to study the behavior of different DVFS governors. They concluded that the interactive governor is faster than on-demand to adapt to the workload changes and thus achieves better performance at about the same energy consumption. Gerards *et al.* [5] analyze the use of global DVFS in the context of multi-core processors. They proposed a theoretical method to transform the problem of finding an optimal clock frequency on global DVFS systems to a single core problem by using the amount of parallelism of applications. Their main goal is to minimize the energy consumption of nontrivial real-time applications. Kin *et al.* [12] proposed a realistic DVFS performance prediction method and a practical DVFS control policy (eDVFS) that aims to minimize total energy consumption in multi-core processors. Their experimental

results show that eDVFS can save a substantial amount of energy compared with Linux on-demand. Isci *et al.* [9] proposed to fine-tune the processor's clock frequency by using workload characteristics aim to maintain a chip-level power below a specified budget without degrading the performance significantly.

Other challenging problem that has been studied extensively to improve the performance of parallel applications is **Energy-Aware Load balancing** [11, 16]. However, few works have made some efforts to further improve the energy consumption. Aupy [1] proposed energy-aware scheduling models to schedule tasks under reliability and makespan constraints. They designed and evaluated them using simulations with different heuristics based on the failure probability, the task weights, and the processor speeds. Sarood *et al.* [14] proposed a load balancing strategy that limits the processors' temperatures to reduce the energy spent in cooling and to prevent hot spots. Their results achieved energy savings of up to 63%, with a timing penalty from 2% to 23%. Goel *et al.* [6] proposed a model that uses CPU performance counters and CPU temperature to generate accurate per-core power estimates in real-time. They showed that the model can be used to guide scheduling decisions in power-aware resource managers. Hartog *et al.* [7] studied the relationship between CPU temperature and energy consumption in clusters and provided a method of estimating the power consumption of the system. This method was then used to implement a MapReduce framework that can evaluate the current status of each node and dynamically react to estimated power usage without having to rely on readings from expensive power monitoring hardware affixed to each node in the cluster.

As opposed to these works, our approach performs load balancing along with DVFS to improve the performance and to reduce the energy consumption by exploiting residual imbalances of parallel applications. In addition, we also reduce the cost of task migrations, since we only migrate tasks between processors when necessary. The performance and energy efficiency of our approach were also analyzed on a set of real-world application running on top of a real platform without the need of simulations.

3 Energy-Aware Load Balancer (EnergyLB)

Scientific applications generally have tasks with different processing demands. Load balancing approaches have been used to mitigate these imbalanced load by making a better load distribution among the available processors in the system. Load balancing algorithms try to mitigate load imbalance by moving tasks between processors, which incurs in overheads to data collection, decision making and mainly tasks migration, and may affect the application runtime and consequently the total of energy spent. However, the most of the strategies aim only on reducing the execution time. Only a few recent strategies began to use the power demand and energy consumption information in their decision making process.

Thus, when imbalanced applications are executed on parallel platforms, the program only finishes its execution when the most loaded processor completes all its tasks. This way, the total execution time of an application is generally defined by the most loaded processor. This means that all other processors or cores in charge of lightweight tasks finish first and remain consuming energy without performing any actual work for the application.

In the case of iterative applications, energy savings can be achieved if the clock frequency of these underloaded processors or cores is decreased through DVFS in such a way that they can still and their tasks before or at the same time of the most loaded ones. Load balancers can be applied to reduce such imbalances but some residual imbalance may still remain after task migrations, making room for further energy improvements.

In principle, DVFS can be performed mainly at two levels of granularity. Systems featuring

per-chip DVFS use the same power delivery network to reach every core, and consequently, bind each core to the same DVFS schedule. On the other hand, systems that allow *per-core DVFS* control use a separate voltage regulator for each core and therefore allow every core to have an independent DVFS schedule.

Taking these two levels of granularity in mind, we propose two variants of Energy-Aware Load Balancer (Figure 1). The first one, is a centralized approach, called *Fine-Grained EnergyLB (*FG-EnergyLB*)*. It is suitable for platforms composed of few tens of cores that allow per-core DVFS. It considers a flat view of the underlying platform, balancing the load among all cores and performing DVFS on each individual core. When the load imbalance is considerably high, we rely on currently available load balancers to better distribute the tasks among the cores. On the other hand, when migrating tasks is not beneficial, we adjust the clock frequency of underloaded cores individually to save energy. Although this approach allows for a finer tune of the load and frequency, it becomes subject to considerable overheads on platforms having hundreds of cores.

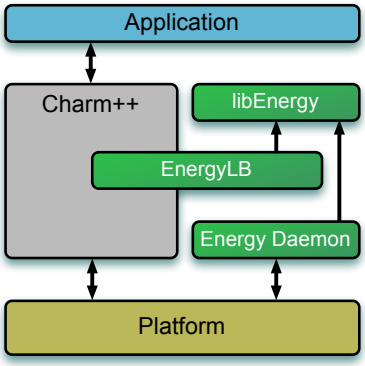


Figure 1: EnergyLB Implementation

To overcome this issue, the second, called *Coarse-Grained EnergyLB (*CG-EnergyLB*)*, uses a hierarchical approach suitable for current HPC platforms, where the underlying platform is represented as a two-level tree. In this scheme, the platform (*root*) is composed of several multi-core processors (*leaves*). Our hierarchical approach performs load balancing to distribute the load among the processor's cores in each leaf. Then, in the root level, it adjusts the frequency of each multi-core processor according to its load to save energy.

Beyond the load balancers, we implemented two other modules. The first, called of Energy Daemon is responsible for gathering power and energy measurements through the Intel's Running Average Power Limit (RAPL). And the second is a Lib Energy that offers an interface to get system information such as the maximum frequency that can be set to a core or processor, to set the clock frequency of individual processors/cores or to set a specific Linux governor.

Our strategies take three input parameters in their execution. The first one, is the load balancer that is used to migrate tasks when the imbalance is high. The second one, is the maximum frequency available by processors that can be set to a core, and the last one, is a threshold value, used to decide whether call the load balancer or perform DVFS strategy.

In this way, in each call of the load balancer, is verified if the weighted load of each processor exceeds or not the *threshold*, make decisions to adjust the frequencies (determining so that the frequency will be decreased or increased) or invoke other load balancer to migrate tasks.

We used the Charm++ parallel runtime system [10] to implement the two variants of

EnergyLB. This framework provides a simple API that can be used to implement load balancers without changing the source code of the applications. The instrumented data load can be dynamically obtained during the application execution through the use of the API.

4 Evaluation Methodology

4.1 Experimental Environment

Our experiments were conducted on an Altix UV 2000 platform designed by SGI. The platform is composed of 24 NUMA nodes. Each node has an Intel Xeon E5-4640 Sandy Bridge-EP x86-64 processor with 8 physical cores running at 2.40 GHz. There are 14 clock frequency levels available in this processor, allowing us to vary the clock frequency of the processor from 1.2 GHz up to 2.4 GHz.

Each core of the Intel Xeon E5-4640 has 32 KB instruction and 32 KB data L1 caches and 256 KB of L2 cache. Each node has 32 GB of DDR3 memory, which is shared with other nodes in a cc-NUMA fashion through SGI's proprietary NUMAlink6. Overall, this platform has 192 physical cores and 768 GB DDR3 memory.

The platform runs an unmodified SUSE Linux Enterprise Server operating system with kernel 3.0.101-0.29 installed. All applications as well as the Charm++ programming model were compiled with GCC 4.8.2. The Charm++ version used in our experiments was linux64-6.5.1. The results presented in Section 5 are the average of at least 10 runs. The relative error was less than 5% using a 95% statistical confidence by Student's t-distribution.

4.2 Applications

To evaluate the runtime, power demand and total energy consumption of our Energy-Aware Load Balancers, we selected three real-world applications. They were chosen due to their varied range of communication patterns and workload characteristics. The description of the applications is given below:

- ***Ondes3D*** is a seismic wave propagation simulator employed to estimate the damage in future earthquake scenarios [3]. In *Ondes3D*, seismic waves are modeled as a set of elastodynamics equations. These equations are then solved by applying a finite difference method. The application is over decomposed into multiple virtual MPI processes per core. *Ondes3D* presents load irregularity due to the boundary conditions producing additional work, and load dynamicity from the simulation of waves spreading through space;

- ***Lulesh*** simulates a variety of science and engineering problems require modeling hydrodynamics, which describes the motion of materials relative to each other when subject to forces. The Livermore Unstructured Lagrange Explicit Shock Hydrodynamics (LULESH) application was originally developed as one of the five challenge problems in the DARPA Ubiquitous High Performance Computing (UHPC) program. *Lulesh* solves one octant of the spherical Sedov problem using Lagrange hydrodynamics [2]; and

- ***Lassen*** is a mini-application that explores applications with varying load balance. It simulates the propagation of a wave through as it travels around an unstructured mesh and it works on a 2D and 3D version. The computational load of the mesh is sub-divided into domains, which are assigned to processors. Therefore, this becomes a challenging problem to effectively parallelize since the workload is constantly changing [13].

4.2.1 Input parameters

Table 1 summarizes the characteristics of the applications and parameters used in our experiments. Different load balancing frequencies have been chosen for different applications in order to strike a balance between the benefits of remapping tasks and the overheads of moving tasks and computing a new task mapping.

Table 1: Summary of the input parameters of applications.

Application	FG-EnergyLB Tasks	CG-EnergyLB Tasks	Iterations	LB Frequency
Ondes3D	128	1024	500	20
Lulesh	729	5832	1000	50
Lassen	256	512	550	50

4.3 Load Balancers

We applied our energy-aware load balancer along with different standard load balancers available in Charm++. The rationale behind that is to evaluate the energy savings that can be obtained by exploiting residual imbalances of these load balancers. To compare the power demand and energy consumption results, we selected the load balancers GreedyLB (G) and GreedyCommLB (GC) that have a greedy algorithm, RefineLB (R) and RefineCommLB (RC) that use a limit of task migrations, RandCentLB (Ra) that randomly assigns objects to processors and ScotchLB (S) that map based on the recursive bi partitioning using both the source process graph and the target architecture graph.

5 Experimental Results

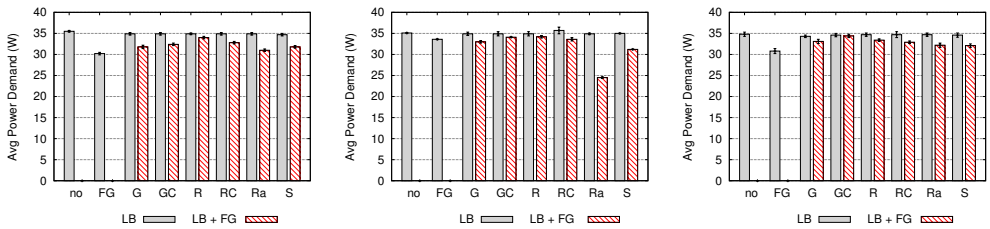
Real applications as *Ondes3D*, *Lulesh* and *Lassen* have different initial imbalance and different dynamicity in their computational load, which leaving space for the use of load balancing algorithms. In this context, some energy improvements were achieved.

Thus, this section provides an analysis of total energy spend with load balancing over total energy consumption and the percentage of energy spend with load balancing when FG-EnergyLB and CG-EnergyLB are used over these real applications. The application runtime depends on several issues, among them, the number of parallel tasks and their load, the duration of each timestep, and the selected load balancing strategy. The impact of load balancing is directly related to the load balancing frequency once load balancing overhead can overcome the gains achieved with load balancing.

5.1 Fine-Grained EnergyLB Evaluation

The propagation of the wave of this application introduce a dynamism to the load. In this way, *Ondes3D* has both irregularity and dynamicity in its execution. Applying FG-EnergyLB alone, it reduces the clock frequency of cores used during the runtime, which reduces the power demand in 15% on average, from 35.5 W to 30.2 W. By reducing the processors power according their computational load, FG-EnergyLB achieve saving energy of 13.9%.

Similar to *Ondes3D*, *Lulesh* and *Lassen* have imbalance and dynamicity of load. Considering this, FG-EnergyLB make several DVFS to adjust the clock frequency of the cores according



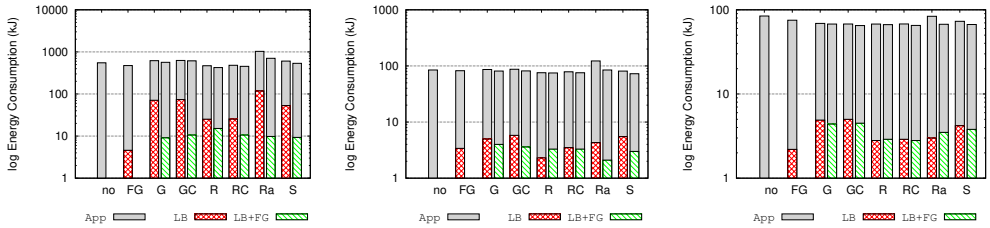
(a) *Ondes3D*. (b) *Lulesh*. (c) *Lassen*.

Figure 2: Average Power Demand reductions for each real application with FG-EnergyLB.

their relative load, reducing the average power of 35.06 W to 33.64 W, to *Lulesh* and of 34.8 W to 30.8 W (11.48%).

When FG-EnergyLB was applied over the residual imbalance of other load balancing algorithms the average power demand is reduced in up to 5.29%, 4.05% and 5.23% respectively.

Considering this reductions in average power demand, we select the energy spent with load balancing when load balancers are used alone (LB) and the energy spent with load balancing when load balancers are used with FG-EnergyLB (LB+FG) of the total energy spent in the application execution (App), as depicted in the Figure 3.



(a) *Ondes3D*. (b) *Lulesh*. (c) *Lassen*.

Figure 3: Energy spend with load balancing over total energy consumption for each real application with FG-EnergyLB.

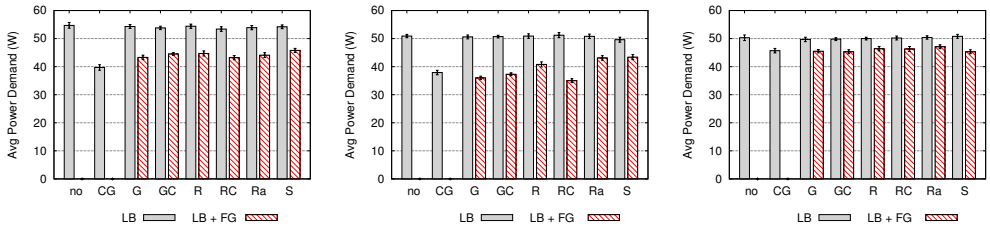
The energy spent with load balancing comes from of the time overhead of every load balancer call. In the test with *Ondes3D*, the load balancers are responsible in average for spending of 7.87% of the total energy consumed during execution of the application (red column in Figure 3(a)). When FG-EnergyLB was used together with load balancers (green column), it is able to saving energy of own load balancing. The energy spent with load balancing was reduced from 7.87% to 2.07%, on average. This occurs due the reduction of the total execution time and mainly by reductions on average power demand.

Over *Lulesh* and *Lassen* applications, the load balancing is responsible for 4.91% and 3.57% respectively of the total energy consumed during execution of the application. Using FG-EnergyLB together with other load balancers, the total energy spent with load balancing is reduced to 3.52% and 3.13% respectively, as shown in the Figure 3(b). In this way, using FG-EnergyLB together with other load balancers, the average of energy spent with load balancing is reduced from 5.45% to 2.90%.

5.2 Coarse-Grained EnergyLB Evaluation

This section provides a CG-EnergyLB evaluation over real-world applications. Each one these scientific applications present a different and dynamic behavior, the load of its tasks change through the iterations, which provides a more challenging scenario for energy aware load balancing. Since all the 192 cores will be used, different parameters are used in the evaluation of the CG-EnergyLB, as shown in the Table 1.

These applications has irregularity in the creation of its tasks and load dynamicity during execution. So, when load balancing algorithms are applied over real applications, different improvements are achieved. By updating the frequency of the processors according to their weighted loads, When CG-EnergyLB is employed alone, it reduces the average power demand of the system while the application is running. Using this approach, reductions of 27.4%, 25.5% and 9.17% on average in power demand are achieved, as depicts the Figure 4.



(a) *Ondes3D*. (b) *Lulesh*. (c) *Lassen*.

Figure 4: Average Power Demand reductions for each real application with CG-EnergyLB.

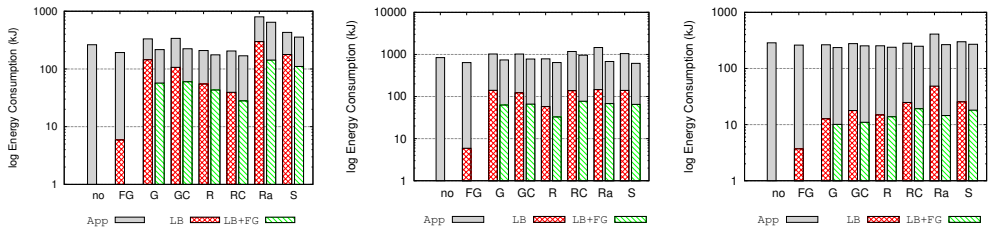
Over *Ondes3D*, CG-EnergyLB (Figure 2(a)) reduces the power from 54.71 W to 39.72 W. For *Lulesh* and *Lassen* (Figures 2(b) and 2(c)), CG-EnergyLB is able to reduce the average power from 50.91 W to 37.93 W and from 50.34 W to 45.72 W.

Once applied load balancers, residual imbalances still are present. In this way, when CG-EnergyLB is employed to mitigate this residual imbalance left by the load balancing algorithm, it identifies the possibility of reducing the processors clock to achieve better gains over these algorithms. For theses applications, the average power demand is reduced in 21%, 20.6% and 8.83% respectively.

Considering the different power reductions, we analyse the load balancing energy when load balancers are used alone (LB) and the load balancing energy when load balancers are used with FG-EnergyLB (LB+FG). These energy value are depicted in the Figure 5 compared to total energy spent in the application execution (App).

We realize that our CG-EnergyLB has load balancing overhead lower than other state-of-the-art load balancers when tested with real applications. So, when the application call CG-EnergyLB, it spend less time to adjust the clock frequency, which results in a reduction in total execution time and an equivalent saving energy.

In the tests with the *Ondes3D*, *Lulesh* and *Lassen* applications, the load balancers are responsible in average by spent of 28.87%, 9.87% and 6.80% of the total energy consumed during execution of the applications. When CG-EnergyLB was used together with load balancers, it reduces the energy spent with load balancing on average to 24.55%, 8.46% and 5.71% respectively.



(a) *Ondes3D*. (b) *Lulesh*. (c) *Lassen*.

Figure 5: Energy spend with load balancing over total energy consumption for each real application with FG-EnergyLB.

6 Conclusions

The exponential increase in power consumption related to a linear increase in the clock frequency and a higher complexity involved in the processors' design changed the course of development of new processors. Power consumption has become a critical aspect to the development of both large and small scale systems. This concern is now enough to warrant the research on techniques to improve the energy efficiency of parallel applications running on top of HPC platforms.

In this paper, we focused on reducing the energy consumption and power demand of parallel applications without considerably degrading their overall performance. Our main idea was to exploit the existence of residual imbalances on iterative applications to adjust the clock frequency of underloaded cores/processors through DVFS. We proposed two variants of EnergyLB, an energy-aware load balancer and validate both variants in Charm++. Both strategies can be used together with any other load balancer to improve energy efficiency of real world applications.

Experimental results present improvements in energy consumption when our EnergyLB load balancers were used with state-of-the-art algorithms over real-world applications. FG-EnergyLB and CG-EnergyLB are able to reduce the load balancing energy in average of 32.6% and 11.1% respectively.

This work can be extended in different directions. One possibility would be to develop a new load balancer that performs load balancing and DVFS at the same time in each load balancing step. For that, it would be necessary to create a heuristic that take into account the cost of task migrations between cores/processors that operate in different clock frequencies. Another possibility would be to develop a hierarchical energy-aware load balancer that performs task migrations between cores of the same processor and only migrate tasks between processors when needed. In this scheme, only the processors involved in task migrations would need their clock frequencies to be adjusted, reducing overhead of performing DVFS on all processors at each load balancing step. Finally, we also intend to evaluate the benefits of FG-EnergyLB and CG-EnergyLB on other real-world applications and platforms.

Acknowledgements

This work was supported by CNPq, CAPES, FAPERGS and FINEP. The research has received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the HPC4E Project, grant agreement number 689772. Work developed on the context of the associated international laboratory between UFRGS and *Université de Grenoble* - LICIA.

References

[1] Guillaume Aupy, Anne Benoit, and Yves Robert. Energy-aware scheduling under reliability and makespan constraints. In *Proceedings...*, pages 1–10. International Conference on High Performance Computing (HiPC), IEEE Computer Society, 2012.

[2] SS Dosanjh, RF Barrett, DW Doerfler, SD Hammond, KS Hemmert, MA Heroux, PT Lin, KT Pedretti, AF Rodrigues, TG Trucano, et al. Exascale design space exploration and co-design. *Future Generation Computer Systems*, 30:46–58, 2014.

[3] F. Dupros, H. Aochi, A. Ducellier, D. Komatitsch, and J. Roman. Exploiting intensive multithreading for the efficient simulation of 3d seismic wave propagation. In *Proceedings...*, pages 253–260. International Conference on Computational Science and Engineering, IEEE, July 2008.

[4] W. Feng and K.W. Cameron. The green500 list: Encouraging sustainable supercomputing. *Computer*, 40(12):50–55, 2007.

[5] Marco E.T. Gerards, Johann L. Hurink, Philip K.F. Holzenspies, Jan Kuper, and Gerard J.M. Smit. Analytic clock frequency selection for global DVFS. In *Proceedings...*, pages 512–519. Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2014.

[6] Bhavishya Goel, Sally A McKee, Roberto Gioiosa, Karan Singh, Major Bhadauria, and Marco Cesati. Portable, scalable, per-core power estimation for intelligent resource management. In *Proceedings...*, pages 135–146. International Green Computing Conference (IGCC), IEEE Computer Society, 2010.

[7] Jessica Hartog, Elif Dede, and Madhusudhan Govindaraju. Mapreduce framework energy adaptation via temperature awareness. *Cluster Computing*, 17(1):111–127, 2013.

[8] Chung-Hsing Hsu, Wu chun Feng, and Jeremy S. Archuleta. Towards efficient supercomputing: A quest for the right metric. In *Proceedings...*, pages 8–pp. International Parallel and Distributed Processing Symposium (IPDPS), IEEE, April 2005.

[9] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. pages 347–358. International Symposium on Microarchitecture (MICRO), IEEE, 2006.

[10] Laxmikant V Kalé, Eric Bohm, Celso L Mendes, Terry Wilmarth, and Gengbin Zheng. Programming petascale applications with CHARM++ and AMPI. *Petascale Computing: Algorithms and Applications*, 1:421–441, 2007.

[11] L.V. Kalé, Milind Bhandarkar, and Robert Brunner. Load balancing in parallel molecular dynamics. In Alfonso Ferreira, José Rolim, Horst Simon, and Shang-Hua Teng, editors, *Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 251–261. Springer Berlin Heidelberg, 1998.

[12] Shin-gyu Kim, Hyeonsang Eom, HeonY. Yeom, and SangLyul Min. Energy-centric DVFS controlling method for multi-core platforms. In *Proceedings...*, pages 685–690. High Performance Computing, Networking, Storage and Analysis (SCC), IEEE Computer Society, Nov 2012.

[13] Brian McCandless. Lassen mini-app, 2015. Accessed: 2015.30.09.

[14] Osman Sarood, Esteban Meneses, and Laxmikant V. Kalé. A 'cool' way of improving the reliability of HPC machines. In *Proceedings...*, pages 58:1–58:12. International Conference on High Performance Computing, Networking, Storage and Analysis (SC), ACM, 2013.

[15] Vasileios Spiliopoulos, Akash Bagdia, Andreas Hansson, Peter Aldworth, and Stefanos Kaxiras. Introducing DVFS-management in a full-system simulator. In *Proceedings...*, pages 535–545. Modelling, Analysis & Simulation of Computer and Tel. Systems (MASCOTS), 2013.

[16] Gengbin Zheng, Abhinav Bhatelé, Esteban Meneses, and Laxmikant V Kalé. Periodic hierarchical load balancing for large supercomputers. *International Journal of High Performance Computing Applications*, 25(4):371–385, 2011.