GreenHPC: A Novel Framework to Measure Energy Consumption on HPC Applications

Gustavo Rostirolla,
Rodrigo da Rosa Righi,
Vinicius Facco Rodrigues
Applied Computing Graduate Program
Unisinos – Av. Unisinos, 950
São Leopoldo, RS, Brazil
Email: rostirolla@unisinos.br
rrrighi@unisinos.br
vfrodrigues@unisinos.br

Pedro Velho
INRIA - Univates
2004, route de Lucioles BP93
06902 Sophia Antipolis CEDEX
Email: pedro.velho@inria.fr

Edson Luiz Padoin
Department of Exact
Sciences and Engineering
Regional University of Northwest of
Rio Grande do Sul (UNIJUI) –
Ijui, RS – Brazil
Email: padoin@unijui.edu.br

***Abstract*—Energy consumption on systems that have a continuous power source is tightly-related to both the computing time of an application and its required CPU load. Considering the scope of HPC applications which commonly have a time precision in nano or milliseconds, we observe a lack of systems that combine appropriate sampling rate, low intrusiveness and low cost. In this context, this article presents a model called GreenHPC that uses a hall effect sensor to precisely capture current with an arbitrary timeslice on HPC applications. Its scientific contribution relies on analyzing the energy consumption at a cluster scale, without application intrusiveness, showing the impact of maintaining idle nodes or turning them off for energy saving. Furthermore, considering the use of GreenHPC over the execution of a seismic wave application, we also present the number of employed processors which present the best energy consumption index. Finally, we have used the obtained results to infer a model to estimate energy consumption of HPC applications. All the developed work has a special concern on reproducibility, so all data and hardware schematics are available for download at [1].**

***Keywords*—High performance computing, green computing, cluster, energy-consumption, measuring tools and methods, ARM**

I. Introduction

Advances in science depend on huge scientific experiments, such as the LHC (Large Hadron Collider) [1], requiring more and more processing power and data storage. Aiming at offering both capabilities, clusters with thousand of nodes are assembled as mainstream architecture for high performance computing (HPC), which leads to unfeasible energy consumption rates of several gigawatts. Thus, researchers now face the challenge on designing HPC systems with energy constrains in mind, so a new branch in computer science called GreenComputing is getting space among consolidate journals and conferences [2]. In this way, both Top500 [3] and Green500 [4] are concern in presenting energy efficiency metrics besides the usual computing rates.

Measuring energy consumption is indeed a subject of research on battery-powered devices. Naturally, the idea of porting this measuring equipments is valid on the HPC domain. The state-of-the-art presents some alternatives on measuring energy consumption as follows: (i) use of TDP (Thermal Design Power) to estimate consumption in a theoretical way [5], [6] and; (ii) rely on hardware sensors such as multimeters [7], [8] or commercial power meters [9], [10], [4]; (iii) analysis of battery life when using mobile devices. The diversity of measuring tools and rigs rise the following problem statement: *Considering the HPC scope, Are we measuring energy consumption in a proper way?*

Particularly, considering the third aforementioned alternative, changes in the order of 0.1W are probably unperceived on a power supplier while in a battery powered device represents a significant percentage of battery life. Besides this comparison, we also observe other two inappropriate methodologies on measuring energy consumption in the HPC landscape: capturing not enough samples in an unbecoming rate (frequency). Sampling rates are especial important for capturing applications power consumption patterns. In HPC, the main objective is to estimate the effective energy cost on using a cluster or grid infrastructure. This is important for a feasibility analysis on maintaining or yet, enlarging, such environments. This study is yet more important when involving multiple users that must pay for using a parallel machine.

In the best of our knowledge, we observe a lack on attacking the duet energy consumption measuring and HPC systems, especially when taking into account sampling rate, cost and intrusiveness. In this context, here we present **GreenHPC** - an affordable and versatile framework for measuring energy consumption on HPC environments with sampling rates up to 500 kSamples/s. This high sampling rate allows a fine grain observation of the application's behavior, and a subsequent analysis of both possible bottlenecks and which sections consume more energy. GreenHPC was designed for not imposing any CPU overhead in the processing nodes, as found in algorithms for estimating power consumption [11], and does not rely on operational systems like Joulemeter [12]. Furthermore, the proposed framework is suitable to measure power consumption in ARM-based processors, which refer to a possible platform that is being considered as candidate to obtain the Exascale computing [5].

Considering distributed systems like clusters and grids, GreenHPC also addresses the distributed clock synchronization

[1] http://grostirolla1.github.io/greenhpc/

978-3-901882-70-8

problem to offer an accuracy visualization and data storage. Aiming at evaluating GreenHPC, we developed a prototype that was tested against two scientific applications (seismology domain and High Performance Linpack) on a cluster featuring 10 ARM Cortex-A8 (Sitara AM3358) computing nodes. The applications were modeled to use from 1 up to 10 nodes and the tests consider as powered on either only the employed resources or the whole configured infrastructure. In this way, besides technical contributions regarding sampling, intrusiveness and clock synchronization, GreenHPC also brings two answers as scientific contributions: (i) *Which is the energy consumption impact on turning off those nodes that are not really being used for application execution?* (ii) *How can we estimate a model to inform the number of nodes that present the best efficiency on energy consumption?* Although considering only two applications, we can extend the methodology for CPU-bound HPC applications without loss of generality.

The remainder of this article will first introduce the related work in Section II. In Section III we review the power consumption basic concepts. Section IV describes GreenHPC in detail. In Section V, we present the conducted experiments using this GreenHPC. In addition, in Section VI we address the results obtained from the experiments. Finally, Section VII emphasizes the scientific contribution of the work and notes several challenges that we can address in the future.

II. Related Work

This section presents our compilation on both methodologies and tools for measuring energy consumption. The set of works were organized in a chronological order. At the end of this section, we present a comparison of the tools based on relevant criteria for HPC: sampling rate, precision, intrusiveness, and cost. Bunse *et. al.* [13] analyze the energy consumption, the execution time, and the required CPU cycles to sort integer arrays of variable size on ATmega16, ATmega32, and ATmega128 processors. The measuring environment relies on a resistor to compute power assuming constant voltage. Moreover, the employed approach is intrusive and requires galvanic decoupling. The experiments show that the *Insertion Sort* - complexity of $O(n^2)$ - has less energy consumption than *Heapsort Sort* - complexity of $O(\log n)$. More precisely, a follow up paper [14] propose models to identify the energy consumption behavior of sorting algorithms.

Pang *et. al.* [7] compare the energy consumption of an ARM cluster against the energy consumption on traditional Intel desktop processors. To accomplish this, the authors have used on ARM processors an equipment named *monsoon power meter*, and another for current sensor clip to get data on Intel processors. Each device has a distinct sampling rate and the results ignore the energy consumption of cooling equipment. Aroca *et. al.* [8] focus on green datacenters, but also observing the two kind of aforementioned processors. They analyze the CPU load, instantaneous power, and temperature in function of the number of handled requests. To measure energy consumption they used a multimeter.

Yi-Cheng *et. al.* [10] propose a low power consumption Database Management System (DBMS) named E2-DBMS. To improve the E2-DBMS energy efficiency, they change Dynamic Voltage Frequency Scaling (DVFS) and analyze the total energy consumption. The energy consumption analysis mostly relies on simulation calibrated with nominal values for components such as disks, memory module, processor, and cooling. Tsirogiannis *et. al.* [9] evaluate the energy consumption on data structures such as *hashes* and *B-Trees*. Results show that the lower the execution time the lower the power consumption. The explanation is that hardware already optimizes energy consumption, when this is the case to achieve lower energy is the co-related to achieving the greater performance. To measure energy consumption they use a Brand Electronics CI to measure the total system power. This power meter has $\pm$ 1.5% accuracy, collecting 1 sample per second.

Bergen *et. al.* [15] analyze the power consumption profiles of different applications that require high allocation of CPU and memory. They measured the power consumption of the entire server rack. So, they created power profiles for different jobs with different resource utilization patterns. With this input, they proposed an energy-aware job scheduler. Bergen *et. al.* [15] highlight the need of a more fine grained monitoring tool to determine profiles of individual servers accurately. According to them, this is important to work with adaptivity and reactivity on green configuration parameters.

The selected references have already showed the wide variety of equipments and methodologies used on measuring energy consumption. Table I summarizes a brief description of the initiatives on developing tools for this context, grouping them in accordance with four characteristics. Analyzing this compilation we can observe a lack of a measuring framework or tool that combines high precision sampling, low intrusiveness, and scalability in a low cost solution. Aiming at filling this gap we proposed GreenHPC, described in details in Section IV.

III. Fundamental Concepts

This section will describe the fundamental concepts considering the topics involving GreenHPC. Firstly, the instantaneous power (P) is the throughput of energy delivered on a specific instant. According to Ohm's Law, power is defined as a product of current (I) by voltage (V). Power is generally measured in watts (W) while current is measured in amps (A) and voltage is measured in Volts (V). So, one way to estimate power is to measure current and voltage as presented in Equation 1. The same physics law states that the effective current depends on a resistor, so the current can be estimated connecting a known resistor to the circuit and using Equation 2 to determinate the current. Nevertheless, this approach has a main flaw. The resistor is intrusive because it can drive two poles with different potential requiring a galvanic decoupling to work [13], [16].

$$P = V \times I \tag{1}$$

$$I = \frac{V}{R} \tag{2}$$

Voltage and current samples enable to compute instantaneous power. Instantaneous power over time estimates the overall energy consumption of a system, *i. e.* the amount of energy used to achieve the solution. This metric is also referenced as energy-to-solution [17]. In this way, aiming at estimating energy-to-solution we need to integrate instantaneous power

TABLE I. COMPARISON OF THE MEASURING TOOLS USING 4 CRITERIA SAMPLING RATE, PRECISION, INTRUSIVENESS, AND COST.

Author	Tool	Sampling Rate	Precision	Intrusiveness	Cost
Bergen *et. al.* [15] Dimitris *et. al.* [9] Ou *et. al.* [7] Aroca *et. al.* [8]	power meter	generally 1 sample/s	high	low	hundreds of USD
Bunse *et. al.* [13]	resistor	10∼100 sample/s	high	high	hundreds of USD
Yi-Cheng *et. al.* [10] Bunse *et. al.* [13]	simulation	configurable	low	low	free
Yi-Cheng *et. al.* [10]	nominal values	-	low	low	free

over time (see Equation 3). In practice we use the discrete time equivalent of the integral using the power and time samples, as Equation 4 illustrates, where for each sample i there is an instantaneous power and a time interval Δt. Figure 1 shows the energy consumption samples over time.

$$E = \int P(t)\, dt \tag{3}$$

$$E = \sum_{i=1}^{N} P(i) \times \Delta t(i) \tag{4}$$

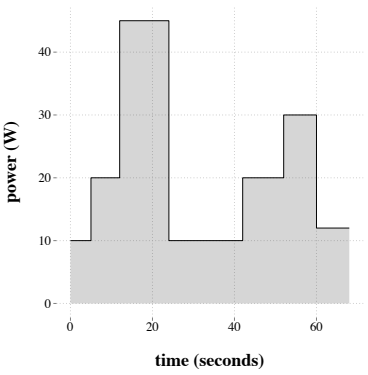


Fig. 1. An example of power samples over time, the energy-to-solution is the definite integral (grey area), in this case 1280 J (joules).

Hereafter we analyze results using two metrics defined previously in [17]:

- **Energy-to-solution** the amount of energy using a discrete interval of power samples over time in joules;
- **Time-to-solution** the amount of time to achieve useful output from the application in seconds.

To measure energy-to-solution we use GreenHPC. Time-to-solution is trivially measured with an in application timer. Our goal is to achieve the result in the lowest time possible spending the less energy possible. We evaluate an HPC environment or application in terms of both to find patterns that lead to best usage, *i. e.* lower scores are better on both metrics.

IV. GREENHPC: LOW COST AND SCALABILITY ON MEASURING ENERGY CONSUMPTION

This section describes GreenHPC - a flexible and low cost framework for power measurement on HPC environments. Figure 2 illustrates GreenHPC components, emphasizing how the power consumption data is collected, stored, visualized and what are the available configuration parameters. The proposed framework has three main components: (i) a Sensor Board which is responsible by current sensing and noise filtering; (ii) a data acquisition board which collects the Sensor Board data and the voltage from the power source; (iii) a Virtual Instrument (VI) which is responsible for the data processing, visualization and distributed clock synchronization.

The scalability of GreenHPC is limited by the Sensor Board, not implying any previous instrumentation in the parallel application to get data about energy. In this way, we can classify the GreenHPC's intrusivity as zero at the programmer's perspective, being independent of both HPC hardware platform and programming language. Although GreenHPC was firstly designed to clusters, it is extensible to other HPC systems such as computational grids. In the last, each cluster of the parallel architecture must have the three aforementioned components and generate log files locally. Upon application events data is collected, GreenHPC applies clock synchronization (for instance, using the Network Time Protocol[2]) and presents energy data in a single graph. Contrary to the *post-morten* solution for grids, GreenHPC on clusters can offer real-time graphs since the framework does not need any network communication.

We used the concepts from GreenHPC framework to develop a simple prototype, as depicted in Figure 3. Technically, the employed Sensor Board is composed by a linear Hall effect-based sensor ACS712ELCTR-30A-T [18] with precision of 66 mV/A and a current range from 0 to 30 A. These parameters provide scalability to monitor energy data with a low cost from a single computer up to a collection of computer racks (with thousands of nodes), as demonstrated by Bergen *et. al.* [15]. Figure 4 depicts the employed Sensor Board schematic in detail. By default the sensor output starts in half of the input voltage in order to measure AC (Alternated Current). Considering that the testbed platform is supplied by DC (Direct Current), a voltage divider and operational amplifiers were used to set the start value to zero. Moreover, this board contains an eight-order Butterworth low-pass filter implemented with a Sallen-Key topology with frequency cutoff of 800 Hz in order to eliminate possible noise arising from the switched-mode power supply.

The Sensor Board data is collected by a National Instruments data acquisition (here named DAQ) board denoted PCIe-

[2]http://www.ntp.org

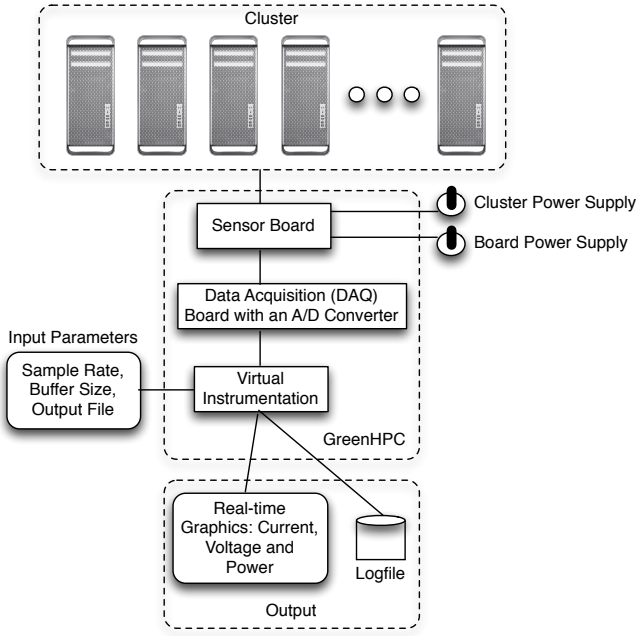


Fig. 2. GreenHPC framework components. The Sensor Board receives two power supplies, where one of them energizes the cluster on computing nodes. The DAQ board acts as an A/D converter, passing digital data of energy consumption to the Virtual Instrument. This last manages voltage, current, and clock synchronization, offering then a zero intrusivity solution at the cluster configuration and user applications perspectives.

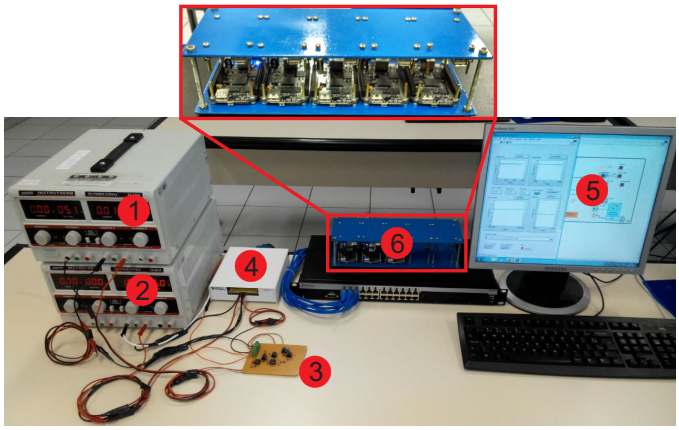


Fig. 3. GreenHPC prototype: (1)Board Power Supply, (2)Cluster Power Supply, (3)Sensor Board, (4)DAQ Board, (5)Virtual Instrument, (6) Cluster of BeagleBone boards, each one with ARM-based processors.

6341 [19]. It has 16 bits Analog-to-Digital Converter (ADC) which presents a sampling rate up to 500kS/s and a timing resolution up to 10 ns. The input range of this DAQ board can vary from 0.2 V with 60 uV accuracy to 10 V with 2.19 mV accuracy. Since the nodes used in the experimental cluster are supplied with $5V$ DC and this value is in the DAQ board input range, the voltage values could be obtained directly by the analog inputs, without needing voltage dividers or optocouplers [16]. The GreenHPC prototype can be used to measure both AC or DC current, alternatively if the voltage is above the DAQ range an optocouplers can be used.

The GreenHPC's third component is a Virtual Instrument (VI), which was developed in the prototype with a tool named

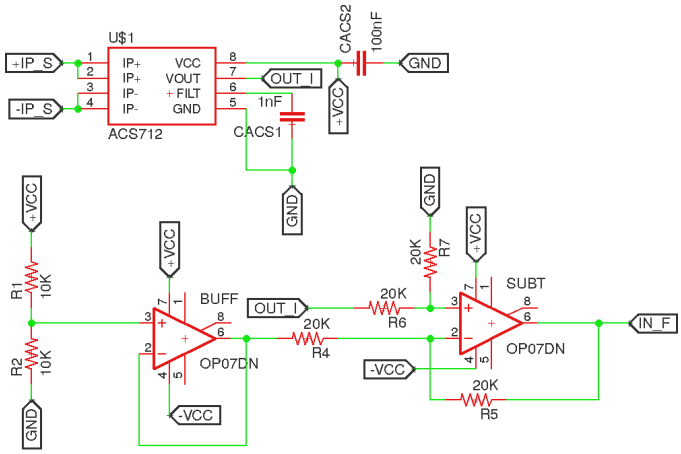


Fig. 4. Current measurement board schematic. The board contains a Hall-Effect base linear current sensor model ACS712 [18] that measures the current and returns a proportional voltage value and a subtracter using operational amplifiers to set the initial value to zero.

LabView [3]. This tool is responsible for the data processing, logging, and real time visualization of the Root Mean Square (RMS) values of voltage (V) and current (A) and also the power consumption (W) calculated using Equation 1. The developed VI provides configurable sample rate and amount of samples that will be stored in the DAQ buffer. The VI also generates an output log file were all the collected samples are stored along with an Epoch timestamp which can be used to calculate the energy-to-solution according to Equation 4.

The division of GreenHPC in three components enables to combine a higher sampling rate, scalability and low intrusiveness. The prototype can bring benefits for programmers and datacenters administrators. Considering the first group, he/she can run his/her application on a thousand of nodes joining then performance peaks and bottlenecks with high-precision energy consumption data. Particularly, this analysis is pertinent when tunning the algorithm complexity. In addition, GreenHPC is agnostic to the employed programming language, middleware or cluster operating system. Concerning the administrators group, they can use GreenHPC to determine the energy consumption of a cluster, and consequently, the prototype can help on decision making about both energy provider selection and infrastructure reconfiguration.

V. Evaluation Methodology

This section presents the platform and the applications used in the tests, besides a brief description on each experiment. The platform is composed by 10 board-based BeagleBone Black computers, each one presenting a 1 GHz ARM Cortex-A8 processor, 512 MByte of main memory, Ubuntu 12.04 operating system with kernel 3.8.13 and gcc compiler version 4.7.3. All the boards were supplied with a switched-mode FA-3050 power supply model from the Instrutherm company[4], and interconnected using a standard Fast Ethernet switch.

Concerning the applications, we are working with: (a) seismic domain model named Ondes3D which simulates the

[3] http://www.ni.com/labview/

[4] http://instrutherm.com.br/fontes-alimentacao-digital-fonte-alimentacao-digital.php

propagation of waves using a tridimensional method of finite differences [20]; (b) High Performance Linpack [5](HPL) which solves a dense linear system using LU factorization. The former was chosen because finite differences method is commonly found on HPC implementations. Particularly, Ondes3D's parameters are defined in a series of files each one having a partial input. They describe the surface properties (such as elasticity and density), and also the selection of the solving method. Other configuration files define the number of points - initial and final - in each direction (x, y, and z axes). However, the main parameter used as workload definition is the number of timesteps, which comprises the number of iterations and then application precision. As output, Ondes3D produces a series of image files that combined generate an animation of the seismic wave propagating on the 3D environment [21]. Finally, second application was selected in order to determine the performance per Watt, which consists of the main metric proposed in Green500 project [22].

The first three experiments where modeled to use the Ondes3D application, while the fourth one employs HPL. All of them are described below:

- Experiment i: This experiment uses a single nodes and varies the number of timestep between 10 and 100 with increments of 10. Its objective is to evaluate the changes on both runtime and energy consumption when varying the number of timesteps. We plan to analyze if it is possible to find a point that represents the best tradeoff between energy consumption and time, or yet, if there is or not a direct relationship between energy consumption and the processing time regardless of the problem size.
- Experiment ii: This experiment varies the number of processing nodes while maintaining a fixed number of timesteps in 100. It aims at evaluating the power consumption of the application in an environment more similar to the ones found in real clusters where the idle nodes are not turned off. Thus, the measuring of energy consumption is performed in all the nodes either they were processing or not.
- Experiment iii: As experiment ii, here we are varying the number of nodes, but measuring energy only on processing nodes. Its objective is to measure the energy saving that could be obtained if the idle nodes of the cluster were turned off.
- Experiment iv: It consists of executing the HPL benchmark both in a unique node and in 10 nodes of the cluster. In order to keep the benchmark running for a minimum of 10 minutes, as recommended in the consumption measurement guide from Green500 [22], two different problem sizes were created. For a single node the array size (Ns) was 5120 and both P and Q, parameters that define the number of processes were equal to 1. For 10 nodes we have an array size (Ns) of 10000, P equal to 1 and Q equal to 10 in order to give a total of 10 processes, and in both cases the number of blocks (NBs) was kept at 128.

[5]http://www.netlib.org/benchmark/hpl/

VI. Results

Aiming at better explaining the results this section is divided in four parts in accordance with the description of the experiments (see Section V). All experiments are evaluated using the aforementioned metrics energy-to-solution and time-to-solution.

A. Experiment i: Single Node

Here we are evaluating both the energy-to-solution and time-to-solution as a function of application workload. Its goal is to understand the relative behavior of energy consumption with respect to time. Figure 5 presents time-to-solution (a) and energy-to-solution (b) as a function of the application workload. As explained earlier, the application generates several output images that simulate the seismic wave propagation. More timesteps characterize the number of images, so the larger the number of supersteps the larger is the computing workload. For each timestep configuration we run the application 10 times. The confidence interval of 95% from a *t-student* distribution is shown at each point sometimes represented as a single stroke when the confidence interval is too small.

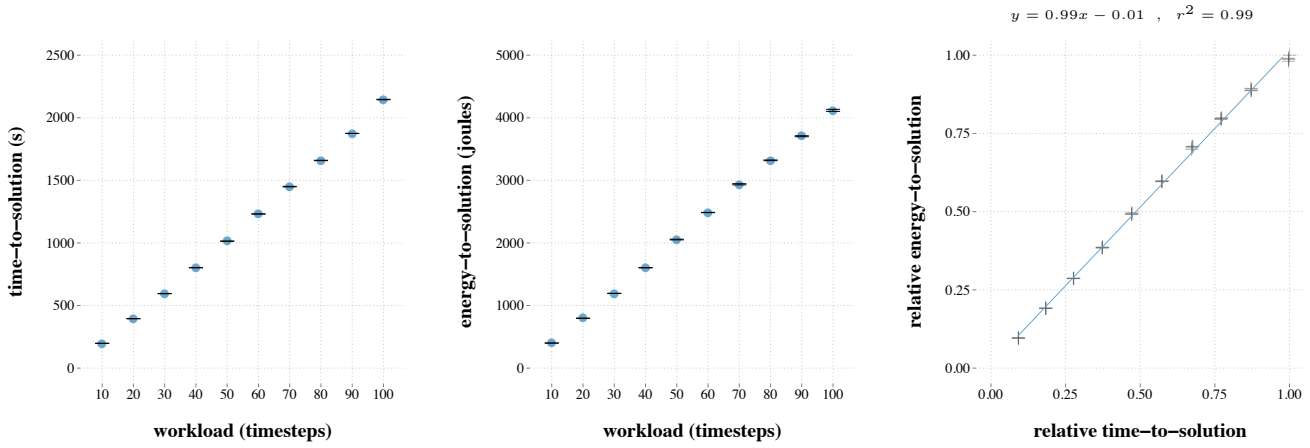
Figure 5 (c) highlights the relationship of energy-to-solution and time-to-solution. The *relative* energy-to-solution is plotted as a function of *relative* time-to-solution, these metrics are relative to the greatest values obtained in Equations 5 and 6. Note that these are all points not only average values as depicted in Figure 5 (a) and (b). We revealed a fitted function model in Figure 5 (c) as being $y = 0.99x - 0.01$. This model has a coefficient of determination $r^2 = 0.99$, which suggests a nearly perfect fit. Therefore, we see that both time and energy consumption grow proportionally, showing the relationship of relative time-to-solution and relative energy-to-solution that reinforces stressful hardware usage. The more time an application takes to execute the more energy it spends. This behavior makes sense since HPC applications are generally stressing the processing units, where the fastest execution is already the best power saving configuration in a single node.

$$relative\ \ consumption = \frac{consumption}{\max(consumption)} \quad (5)$$

$$relative\ \ time = \frac{time}{\max(time)} \quad (6)$$

B. Experiment ii: Multiple Nodes, Maintaining the Whole Cluster Infrastructure Turned on All the Time

In the scope of a single processing unit, more time leads to more energy consumption. In this section we plan to analyze the energy metric scalability when using a typical distributed-based HPC environment. To accomplish this, we compare energy-to-solution and time-to-solution with a variable number of computing nodes. Figure 6 shows the energy-to-solution (a) and time-to-solution (b) when varying the number of processors (one per node). Moreover, we are reproducing the common strategy when using and managing a cluster, that consists of maintaining all nodes as power on independently of their use for jobs computation.

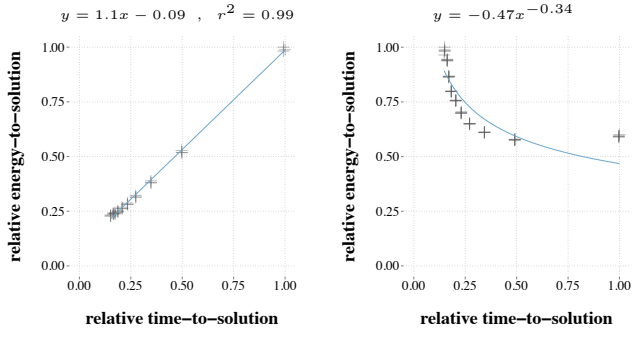


(a) (b) (c)

Fig. 5. Time-to-solution (a) and energy-to-solution (b) as a function of workload (number of timesteps). Time and and energy consumption grows proportionally as seen in (c) that shows energy-to-solution (relative to largest value) as a function of time-to-solution (relative to longest execution) and the fitted linear model $y = 0.99x - 0.01$. The coefficient of determination $r^2 = 0.99$ suggest a near perfect fit. This behavior is expected in HPC applications that stress CPU usage.

The perspective of energy consumption in Figure 6 (a) presents the following idea: the greater the number of employed nodes, the larger the energy saving. Execution time, as depicted in Figure 6 (b), shows a typical behavior of a large-grained distributed application where the time drops as the number of the nodes increases up to a parallelization limit. This degradation behavior means that the amount of time effectively involving computing actions starts to reduced which can be explained by either the grain reduction or synchronization and preconditioning routines that are inherently sequential. The tests here revealed that the effectively use of available resources, *i. e.* near optimal speedups, leads to better energy consumption. Since we are capturing energy data during application execution, the faster its execution, the larger the energy saving.

C. Experiment iii: Multiple Nodes, but Turning Off the Idle Ones



(a) (b)

Fig. 7. (a) represents energy consumption (relative to largest) as a function of time (relative to longest execution) with all resources powered on and (b) with unused resources powered off.

When using a subset of computing nodes of a cluster platform, for instance, a fair question is: *How much energy can we save by shutting down the other cluster nodes?* In this context, Figure 6 (c) and (d) show energy-to-solution and time-to-solution as a function of the processing units. Turning off idle resources indeed present a different behavior on energy-to-solution, in contrast of the situation where all resources were powered on (see this comparison between Figs. 6 (a) and (c)). The energy consumption dropping is significant, where the worst case consumption is below 10 kJ while we have a measure close to 30 kJ when observing Figure 6 (a). In Figure 6 (c), the lowest energy consumption occurs when involving the application execution in a single node. In addition, we observe that when using more nodes we should spend more energy to maintain them powered on, but the parallelism implies on reducing the application time and, consequently, the time to maintaining them powered on as well.

To better understand the relationship of energy consumption and time when turning off idle resources we analyze closely the relative behavior of energy-to-solution as function of time-to-solution on Figure 7. In (a) we see the relative energy versus time on a typical HPC environment where all nodes (including the idle processing units) remain on. Figure 7 (b) present the behavior when idle resources are shutdown. Particularly, the energy consumption drops to a limit close to 0.52 of relative energy-to-solution. Near to the best of saving configuration, the time penalty does not necessarily imply on a proportional energy consumption gain. For instance, energy consumption with 9 processors is 6.5 kJ. With 8 processors (since we power off idle resources) the energy consumption drops to 6 kJ, thus saving about 8% of energy (see these data in Figure 6 (c)). Comparing this same number of processors, the time penalty is about 4%, 349.24 seconds with 9 processors that rise to 364.51 seconds with 8 processors. Clearly, there is a gap on improving the time versus energy consumption trade-off.

On Figure 7 (a) and (b) we also have two fitted function models that describe the aforementioned behavior. In the first case, a linear model best fits the behavior, while an inverse logarithmic behavior suites the second case. This logarithmic behavior cope with the time-to-solution drop as application scale in comparison to a single processor: the ideal case is hal f the time using 2 processors, a third of the time using 3 processors, and so on. Many HPC applications have a non-

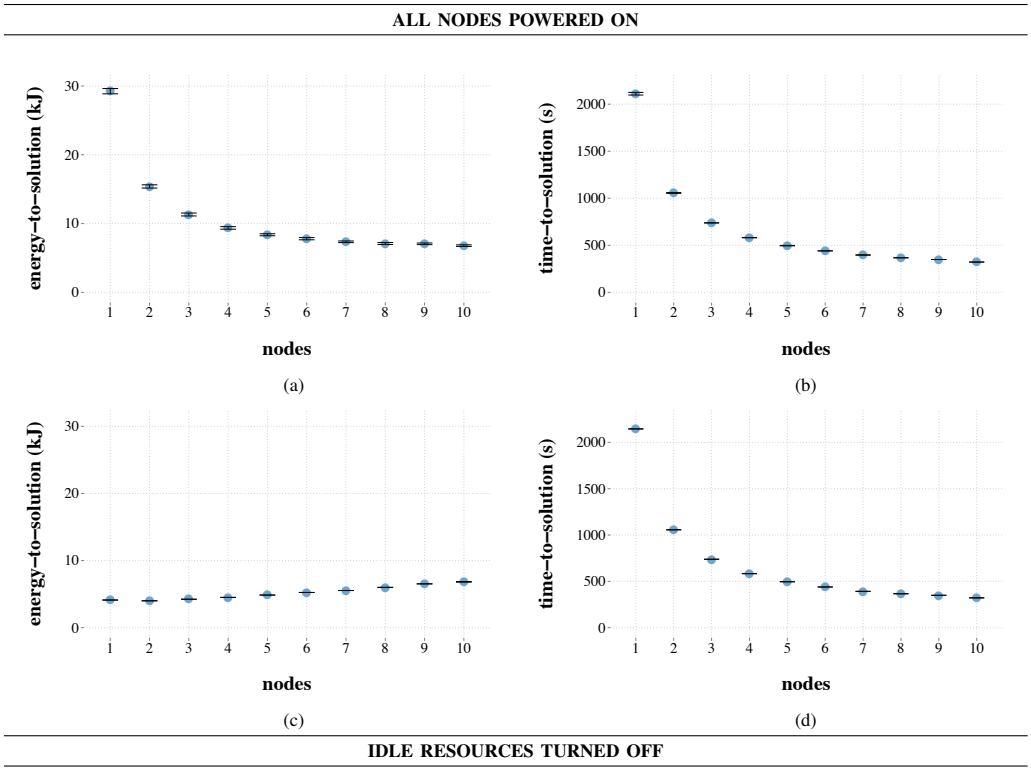


Fig. 6. Energy consumption (a) and computing time (b) as a function of the processing units with all resources powered on and turning off idle resources (c) and (d). The points are mean values over 10 executions. Assuming a 95% t-distribution we present the confidence interval at each point sometimes represented as a single stroke when the confidence interval is too small. Here we numerically demonstrate that the action of turning off idle resources implies on a lower energy consumption.

parallelizable portion, so due to this reason the speedup falls apart from the ideal. Amdahl's law explains this phenomenon as the system optimization is only perceived on the total portion of the system which is really optimized. Hence if the parallel application has a sequential portion preconditioning algorithms or synchronization points, for instance, the gain on time-to-solution degrades. We hence propose another model that can better explain this behavior.

Based on the shape of Figure 6 (c) energy-to-solution grows linear when turning off idle resources. Meanwhile the time-to-solution, Figure 6 (d), follows an inverse logarithmic drop. The relative time-to-solution of the application is never less than 0.16 and the amount of relative energy should be at least 0.52 (energy of having one board on). This leads to the empirical model of Equation 7 plotted aside with the data on Figure 8. In this model we see the faster execution as a constant that indicates the optimal relative time-to-solution being equal to 0.16. Energy-to-solution has also a lower bound on 0.52, which is the single processing unit configuration, *i. e.* the most power saving one. This last can also adapt to other scenarios. The constant -0.04 explain the curve shape of relative energy-to-solution versus time-to-solution.

$$y = -0.04 \log_2 (x - 0.16) + 0.52 \tag{7}$$

On the first two experiments (Subsections VI A and B), the energy consumption follows a linear approach, where a faster execution time leads to a lower energy-to-solution and the opposite is also true (lower execution time leads to higher energy-to-solution). On the other hand, when unused resources are shutdown (see Figure 7 (b)), this behavior changes and

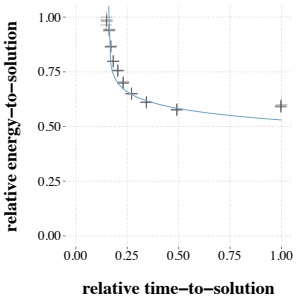


Fig. 8. Empirical model describing the relative energy-to-solution as a function of relative time-to-solution.

faster execution become the more power hungry strategy. The less processors we use the more energy we save but this implies on a execution time penalty. Here lies a central question: *which is the optimal point of energy consumption versus execution time?* Our model goals to ease the decision process on that matter however time-to-solution and energy-to-solution constraints also depend on the application goal. For instance, on cloud computing costumer and infrastructure provider sign a Service Level Agreement (SLA) that guarantee a certain Quality of Service (QoS). On these cases game dynamics could reward users that cope with energy constraints.

D. Experiment iv: Operations per Watt

This subsection presents the energy efficiency as how many floating point operations per spent watt our system can achieve. This metric complies with several lists available on the Internet such as Green500 and Top500. Table II shows the results when using 1 and 10 nodes. Considering the Mflops analysis, we can

affirm that HPC produces a sub-linear speedup equal to 8.2. This sentence helps to explain the lower performance per Watt obtained when employing 10 nodes. Network communication and barrier synchronizations of the HPC application explain the gap of 12% of this metric for 1 and 10 nodes.

TABLE II. HIGH PERFORMANCE LINPACK RESULTS COMPARISON

	1 Node	10 Nodes
Performance (Mflops)	67.066	551.9
Energy (kJ)	2.5	24.6
Time (s)	1355.83	1219.25
Average Power Consumption (W)	1.847	20.191
Performance Per Watt (Mflops/Watt)	36.31	27.334

VII. CONCLUSION

Measuring energy consumption on HPC platforms is still a challenge, where many alternatives rely on a resistor that require galvanic separator (which is very expensive) or present a poor sampling rate. Thus, here we proposed GreenHPC as an alternative that combines appropriate sampling (up to 500k samples per second), low cost, AC/DC compatibility, user application and operating system diagnostic, and non-intrusiveness on analyze the power of the cluster's nodes. Today, GreenHPC is focusing on clusters, but it can be employed to capture data on highly-large distributed systems like computational grids. Each node will generate a single log file and then GreenHPC works with clock synchronization to combine these data in a single visualization graph about the energy consumption.

Our evaluation on a single node agreed that the lower the computational time the lower the energy consumption. Considering the distributed fashion, the results pointed out the benefits of turning off the idle resources. Ideally we can conclude that the best energy-to-solution and time-to-solution trade-off appears as soon as the speedup saturates. This last conclusion can give much insight for cloud economy patterns. Today, our research efforts in providing multilevel sensoring of energy consumption, giving support for HPC developer to understand power savings on several levels: on the power outlet, on hardware modules, between outlet and computing node.

ACKNOWLEDGMENT

This paper was partially founded by Santander and the following Brazilian agencies: CNPq, CAPES, FAPERGS.

REFERENCES

[1] P. B. Lyndon Evans, "LHC Machine," *Journal of Instrumentation*, vol. 3 (08), 2008.

[2] O. Villa, D. R. Johnson, M. O'Connor, E. Bolotin, D. Nellans, J. Luitjens, N. Sakharnykh, P. Wang, P. Micikevicius, A. Scudiero, S. W. Keckler, and W. J. Dally, "Scaling the power wall: A path to exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 830–841. [Online]. Available: http://dx.doi.org/10.1109/SC.2014.73

[3] Top500, "Rank of 500 Fastest Supercomputing Sites," http://www.top500.org/, 2013.

[4] W. chun Feng and H. Lin, "The green500 list: Year two," in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, April 2010, pp. 1–8.

[5] MontBlanc, "Project Website," http://www.montblanc-project.eu/, 2013, [Online; visited in May 2014].

[6] E. L. Padoin, D. A. G. Oliveira, P. Velho, and P. O. A. Navaux, "Evaluating the Performance and Energy of ARM Processors for High Performance Computing," in *41st International Conference on Parallel Processing (ICPP), 1st International Workshop on Unconventional Cluster Architectures and Applications (UCAA)*. IEEE, 2012, pp. 1–8.

[7] Z. Ou, B. Pang, Y. Deng, J. Nurminen, A. Yla-Jaaski, and P. Hui, "Energy-and Cost-efficiency Analysis of ARM-based Clusters," in *12th IEEE/ACM Cluster, Cloud and Grid Computing (CCGrid)*, May 2012, pp. 115–123.

[8] R. Aroca and L. G. Gonçalves, "Towards Green Data-Centers: A Comparison of x86 and ARM Architectures Power Efficiency," *Parallel and Distributed Computing*, vol. 72(12), pp. 1770–1780, 2012.

[9] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *ACM SIGMOD*, 2010, pp. 231–242.

[10] Z. Xu, Y.-C. Tu, and X. Wang, "PET: Reducing Database Energy Cost via Query Optimization," in *PVLDB*, vol. 5(12), 2012, pp. 1954–1957.

[11] D. Versick, I. Waß mann, and D. Tavangarian, "Power consumption estimation of CPU and peripheral components in virtual machines," *ACM SIGAPP Applied Computing Review*, vol. 13, no. 3, pp. 17–25, Sep. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2537728.2537730

[12] A. Kansal, F. Zhao, and J. Liu, "Virtual machine power metering and provisioning," *Proceedings of the 1st ...*, pp. 39–50, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1807136

[13] C. Bunse, H. Höpfner, E. Mansour, and S. Roychoudhury, "Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments," in *Mobile Data Management*, 2009, pp. 600–607.

[14] C. Bunse and H. Höpfner, "Towards an Energy Aware DBMS - Energy Consumptions of Sorting and Join Algorithms," in *21. GI-Workshop on Foundations of Databases*, 2009, pp. 69–73.

[15] A. Bergen, R. Desmarais, S. Ganti, and U. Stege, "Towards software-adaptive green computing based on server power consumption," *Proceedings of the 3rd International Workshop on Green and Sustainable Software - GREENS 2014*, pp. 9–16, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2593743.2593745

[16] F. Englert, T. Schmitt, S. Kossler, A. Reinhardt, and R. Steinmetz, "How to auto-configure your smart home?: High-resolution power measurements to the rescue," in *Proceedings of the Fourth International Conference on Future Energy Systems*, ser. e-Energy '13. New York, NY, USA: ACM, 2013, pp. 215–224. [Online]. Available: http://doi.acm.org/10.1145/2487166.2487191

[17] E. Padoin, D. de Oliveira, P. Velho, and P. Navaux, "Time-to-solution and energy-to-solution: A comparison between arm and xeon," in *Applications for Multi-Core Architectures (WAMCA), 2012 Third Workshop on*, Oct 2012, pp. 48–53.

[18] *ACS712-Datasheet*, Allegro MicroSystems Inc., 2007, rev. 7. [Online]. Available: http://www.allegromicro.com/~/media/Files/Datasheets/ACS712-Datasheet.ashx

[19] *NI 6341/6343 Specifications*, National Instruments, Jan. 2013. [Online]. Available: http://www.ni.com/pdf/manuals/370786d.pdf

[20] H. Aochi, T. Ulrich, A. Ducellier, F. Dupros, and D. Michea, "Finite difference simulations of seismic wave propagation for understanding earthquake physics and predicting ground motions: Advances and challenges," *Journal of Physics: Conference Series*, vol. 454(1), 2013.

[21] D. Michéa and D. Komatitsch, "Accelerating a Three-dimensional Finite-difference Wave Propagation Code Using GPU Graphics Cards," *Geophysical Journal International*, vol. 182(1), pp. 389–402, April 2010.

[22] T. R. Scogland, C. P. Steffen, T. Wilde, F. Parent, S. Coghlan, N. Bates, W.-c. Feng, and E. Strohmaier, "A power-measurement methodology for large-scale, high-performance computing," in *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '14. New York, NY, USA: ACM, 2014, pp. 149–159. [Online]. Available: http://doi.acm.org/10.1145/2568088.2576795